4

Principles and Algorithms for Causal Reasoning with Uncertainty

Jay Weber

Technical Report 287

May 1989

UNIVERSITY OF
ROCHESTER
COMPUTER SCIENCE

Principles and Algorithms for Causal Reasoning with Uncertainty

by

Jay Charles Weber

Submitted in Partial Fulfillment

of the

Requirements for the Degree

DOCTOR OF PHILOSOPHY

Supervised by James F. Allen

Department of Computer Science

University of Rochester

Rochester, New York

1989

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS
BEFORE COMPLETING FORM

1. REPORT NUMBER 287	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Principles and Algorithms for Causal Reasoning with Uncertainty		5. TYPE OF REPORT & PERIOD COVERED Technical Report 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Jay Weber		8. CONTRACT OR GRANT NUMBER(s) N00014-82-K-0193
9. PERFORMING ORGANIZATION NAME AND ADDRESS Computer Science Department Computer Studies Bldg. 734 University of Rochester, Rochester, NY 14627		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS D. Adv. Res. Proj. Agency 1400 Wilson Blvd. Arlington, VA 22209		12. REPORT DATE May 1989 13. NUMBER OF PAGES 109
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Office of Naval Res. Information Systems Arlington, VA 22217		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution of this document is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

none

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

planning, knowledge representation,
frame problem

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This thesis examines representational issues that arise when reasoning about causes and effects given incomplete and uncertain knowledge about the domain. These issues are largely covered by the famous frame and qualification problems. This thesis shows how traditional nonmonotonic logic approaches can be modified to address these problems in a simple, domain-dependent way. This nonmonotonic approach is then generalized to manipulate statistically-founded beliefs, allowing for more consistent and fine-grained representation of causal knowledge.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. ABSTRACT (Continued)

In addition, this thesis presents an algorithmic approach for efficient parallel computation of statistical predictions. This approach involves two heuristics, highest impact first and highest impact remaining, which control the speed of convergence and error estimation for an algorithm that iteratively refines degrees of belief. This algorithm has been implemented and tested by a program called HITEST, which runs on parallel hardware.

Curriculum Vitae

Jay Weber was born December 27, 1962 in Salisbury, Maryland. He grew up in suburbs of Baltimore, Philadelphia, and Washington D.C. After graduating from high school second in his class, he entered the University of Maryland in 1979 to study computer science. While at Maryland, he was chosen to be a teaching assistant for both regular and honors introductory computer science courses. Also at Maryland, he worked for the Computer Vision Laboratory, writing both systems staff and image processing programs under the supervision of Dr. Azriel Rosenfeld. In his senior year, Jay became interested in Artificial Intelligence through a course taught by Don Perlis. Jay graduated in 1983 with a Bachelor of Science degree in Computer Science with Departmental Honors.

In the fall of 1983, the University of Rochester accepted him into the Ph.D. program in Computer Science. After being a teaching assistant for the *Data Structures* and *Compilers and Interpreters* computer science courses, he became a research assistant for his advisor, James Allen. In the spring of 1985, Jay received a Master's Degree in Computer Science from Rochester, and he continued on towards a Ph.D. Four years later, he produced this thesis which sits before you.

Acknowledgements

This work was supported in part by the Air Force Systems Command, Rome Air Development Center, Griffiss Air Force Base, New York 13441-5700, and the Air Force Office of Scientific Research, Bolling AFB, DC 20332 under Contract Number F30602-85-C-0008 which supports the Northeast Artificial Intelligence Consortium (NAIC). This work was also supported in part by U.S. Army Communication Electronics Command grant no. DAAB10-87-K-022, and ONR/DARPA research contract no. N00014-82-K-0193.

Several of the main points in Chapters Two and Three followed directly from discussions I had with Len Schubert. The adaptation of statistical inference notations found in Chapter Five was a product of numerous discussions with Josh Tenenberg and Henry Kyburg. John Mellor-Crummey helped me get going and stay going on the Butterfly multiprocessor, and Tom LeBlanc helped me use SMP. I have learned much from technical discussions with the "Gang of N" crowd, including Leo Hartman, Rich Pelavin, Henry Kautz, Diane Litman, Steve Whitehead, Hans Koomen, Paul Cooper, Mike Swain, Nat Martin, Pat Hayes, and Fahiem Bacchus.

Thanks to my committee for the creative freedom and generous encouragement they provided, especially to: Jerry Feldman, for fostering the department's atmosphere of competence and cooperation, and for being so perceptive during my six-month reviews; to Henry Kyburg, for his soft-spoken authority and for all the time he donated as I explored the field of statistical inference; and to my advisor James Allen, for his patience, support and penetrating insights during all phases of my stay at Rochester.

Thanks to the fine systems and secretarial staff for all the friendly technical and administrative assistance they have given me over the years. Special thanks to Peggy Frantz for being so cheerful and competent each of the hundred times I asked for help.

Thanks to my parents Ron and Mary Jane, and my grandparents Harry, Jeanette, Preston and Vera, for the love of learning they inspired, the confidence they instilled, and the opportunities they worked to bring.

And finally, thanks to Susan, a fellow computer scientist, my best friend, and my wife. I never knew graduate school would work out so well!

Abstract

This thesis examines representational issues that arise when reasoning about causes and effects given incomplete and uncertain knowledge about the domain. These issues are largely covered by the famous *frame* and *qualification* problems. This thesis shows how traditional nonmonotonic logic approaches can be modified to address these problems in a simple, domain-dependent way. This nonmonotonic approach is then generalized to manipulate statistically-founded beliefs. allowing for more consistent and fine-grained representation of causal knowledge.

In addition. this thesis presents an algorithmic approach for efficient parallel computation of statistical predictions. This approach involves two heuristics. *highest impact first* and *highest impact remaining*. which control the speed of convergence and error estimation for an algorithm that iteratively refines degrees of belief. This algorithm has been implemented and tested by a program called HITEST. which runs on parallel hardware.

Contents

List of Figures

Chapter 1

Introduction

Host: My first guest is someone you've all heard about but probably never met. Let's all welcome Robbie, the hypothetical autonomous agent!

Robbie: Thanks. Great to be here.

Host: Why don't you start off by telling everyone what exactly it is that you do.

Robbie: Well Host, I do many different things: stack blocks, shoot guns, start cars, deliver bagels, but mostly I just think.

Host: You must have a lot to think about.

Robbie: Absolutely. See, I'm supposed to be intelligent, which many people interpret as doing things like people do them. People tend to take their skills for granted, but many of the things I'm suppose to do are really quite subtle.

Host: Such as –

Robbie: Take causal reasoning. I'm supposed to be able to make predictions about the future based on information about the past. Simple, right? Wrong. There are all kinds of problems when you actually get down to details.

Host: I've heard of something called the Frame Problem.

Robbie: Yeah, that's a very popular problem. The standard definition goes something like this: "even if a causal reasoner has some reasonable way to infer when some things change, it still has to have a way of inferring that the rest of the world stays the same". A pretty tall order, really, since the rest of the world is a really "bloomin, buzzin place", and doesn't necessarily want to stay the same.

Host: I don't understand why you need to know about the rest of the world.

Robbie: A causal reasoner has to walk a very fine line. On the one hand, you don't want me to make outrageous claims about what goes on when, and on the other hand you don't want me to be so conservative that I miss some useful predictions.

Host: Tell us about some other problem in causal reasoning.

Robbie: Well, another biggie, maybe the biggest, is called the qualification problem. It's quite old, and I guess you could say it's famous, but there's been relatively little work on it. People seem to have heard of it, but aren't really sure what it is. The folk definition is something like "there will be too many preconditions for a realistic prediction". Now this could be interpreted in a number of ways -

Host: Excuse me for a second Rob, but I thought you might want to say what a "precondition" is.

Robbie: Sure. Preconditions are the facts in the past that jointly imply some effect in the future. Ideally, if you knew that all of some effect's preconditions were true, then you could predict that effect with certainty. The qualification problem says that in general, the reasoner won't be able to know all of the facts necessary to make a certain prediction.

Host: So then do you make uncertain predictions?

Robbie: In a manner of speaking, yes, but usually I am designed to make predictions that look like they are certain, but can later be denied if I find out they are wrong. Trouble is, while they are acting like certainties, they can end up combining and making some outrageous claims. For example, if it is reasonable to assume that a light bulb will be working from one day to the next, and this assumption is treated as certain, then the reasoner can decide that it is also certain that the light bulb will be working for a year, or a decade.

Host: Don't need many light bulbs that way.

Robbie: You still need them, you just don't *know* that you need them!

Host: We're running short of time, maybe you should say something about this new thesis you are in.

Robbie: Good idea. It's called "Principles and Algorithms for Causal Reasoning with Uncertainty". The title says it all, really. The work looks at the problems I've been describing, evaluates the effectiveness of some previous solutions, and then presents some new solutions.

Host: Tell us a little about the plot.

Robbie: Well, it starts off with this silly scene-setter dialogue in Chapter 1. Chapter 2 is somewhat drier, where the author describes some old and new notations for representing the knowledge needed for causal reasoning. This sets up Chapter 3, which talks about the frame problem, and how it should be solved by domain-dependent knowledge about how facts persist, instead of the traditional non-monotonic approach.

Host: Sounds okay so far

Robbie: Then in Chapter 4. the author dives in to the qualification problem, and shows several difficulties with the classic solution using circumscription. This sets a negative tone. but then the author turns it around and talks about what's useful about default logic in general.

Host: And that's it?

Robbie: Oh, no. The plot changes to talk about the use of statistics for the representation of causal knowledge.

Host: And statistics do this better than default logic?

Robbie: They share many common features. but the statistical approach can generate predictions between certainty and falsehood. Not only does this allow for more detailed relationships between facts. it can avoid the problem I mentioned about invincible light bulbs.

Host: So what do you do in the thesis?

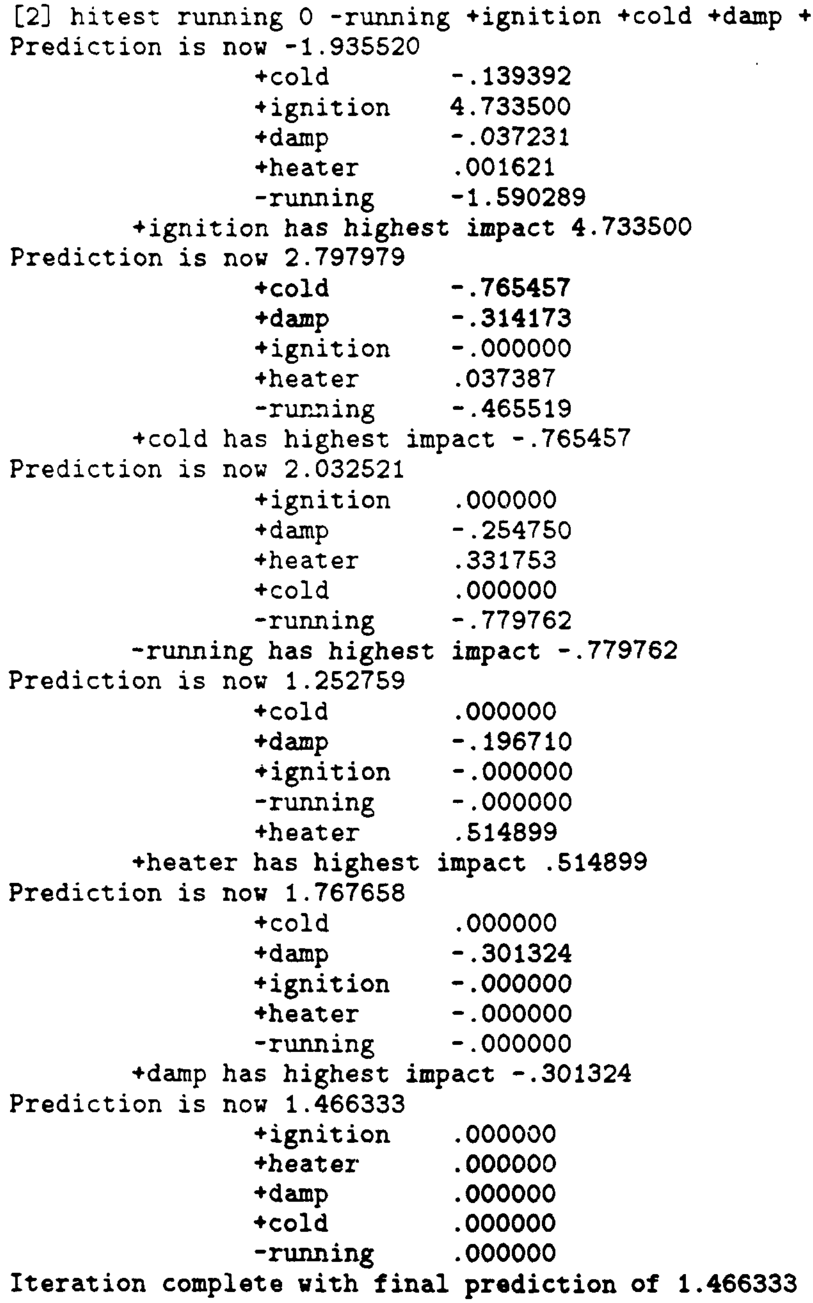
Robbie: I start a lot of cars. or at least I try to. See, the qualification problems says that there are so many things that can go wrong with a car. I can never be *sure* that the car will start. Statistics give me a quantitative way to estimate how likely it is that the car will actually start, given what I know about the situation. and my past experience with starting cars. In Chapter 6. I use the statistical relationships between preconditions and effects to decide which preconditions are more important than others. Therefore. if there are constraints on how much time I have to reason. I can choose to consider the most important preconditions, and ignore the rest.

Host: But surely this ordering of preconditions must take too much time to figure out.

Robbie: That's the real climax of the thesis — Chapter 7 describes how the ordering can be computed in parallel. Each precondition computes its statistical impact on the effect at the same time, and the one with the highest impact is considered first. Then the preconditions all recompute their statistical impacts with respect to this considered fact. The process repeats until nobody has a significant impact, and the current value of the belief is used.

Host: Wow. Do we get a clip from this thesis?

Robbie: Just happen to have one. This clip is a sample run of a program called HITEST that runs on parallel hardware (see Figure 1.1). HITEST is going to compute a belief that the car will be running given that now the car is not



```
[2] hitest running 0 -running +ignition +cold +damp +heater
Prediction is now -1.935520
                +cold         -.139392
                +ignition     4.733500
                +damp         -.037231
                +heater       .001621
                -running      -1.590289
        +ignition has highest impact 4.733500
Prediction is now 2.797979
                +cold         -.765457
                +damp         -.314173
                +ignition     -.000000
                +heater       .037387
                -running      -.465519
        +cold has highest impact -.765457
Prediction is now 2.032521
                +ignition     .000000
                +damp         -.254750
                +heater       .331753
                +cold         .000000
                -running      -.779762
        -running has highest impact -.779762
Prediction is now 1.252759
                +cold         .000000
                +damp         -.196710
                +ignition     -.000000
                -running      -.000000
                +heater       .514899
        +heater has highest impact .514899
Prediction is now 1.767658
                +cold         .000000
                +damp         -.301324
                +ignition     -.000000
                +heater       -.000000
                -running      -.000000
        +damp has highest impact -.301324
Prediction is now 1.466333
                +ignition     .000000
                +heater       .000000
                +damp         .000000
                +cold         .000000
                -running      .000000
Iteration complete with final prediction of 1.466333
```

Figure 1.1: Example run of HITEST program. The fluents are considered one at a time in an order determined by the dynamic values of the fluents' impacts. The numeric values are logarithmic odds; positive values are support for the fluent and negative values are support for fluent's negation.

running. its ignition is engaged. the weather is cold and damp. and the car has a special device that keeps the engine block warm in cold weather. One of these facts will be recognized during each iteration. The new fact is the one with the largest absolute value of its impact. Look how the engaged ignition causes the prediction of the car running to jump from negative (leaning towards not running) to positive (leaning towards running) after the first iteration. The facts are considered one by one in order of statistical importance. finally converging on an answer.

Host: I should point out that Robbie is a professional, and our readers should not try this at home. Any more action in the thesis. Rob?

Robbie: Chapter 8 talks about how degrees of belief can be used to derive other degrees of belief. involving some combinatorial problems and ideas on how to solve them. This is related to the work of Judea Pearl. which you may have seen.

Host: So what's in store for you now that you've gotten better at car starting?

Robbie: Well. I've been thinking about doing some full-fledged planning. involving the manipulation of unusual or unknown objects. or maybe some financial market analysis. involving forecasting and cost/benefit measurement. Before I do, though. more work needs to be done on the incorporation of decision theory. and how to exploit the properties of various statistical distributions. There's been talk about several sequels.

Host: There you have it. Thanks for coming. Rob, and readers: don't miss the rest of the thesis!

Chapter 2

Logical Representations for Causal Reasoning

This chapter reviews some of the more influential representations of knowledge about causality. All of these representations are based on *first-order predicate calculus*, a language for constructing formal theories about problem-solving domains. This thesis assumes that the reader has a basic understanding of the syntax of first-order logic, as well as some exposure to its use in knowledge representation. The is some variation, however, in the symbols used to write sentences. I use the following symbols and meanings:

$\wedge$	conjunction
$\vee$	*disjunction*
$\neg$	negation
$\rightarrow$	material implication
$\forall$	infinite conjunction
$\exists$	infinite disjunction
$\bigwedge$	finite conjunction
$\bigvee$	finite disjunction

Predicate, function, and object constants are printed using the typewriter font ($\texttt{p}$), and variables using math italics (p). Unbound variables are implicitly universally quantified.

2.1 The Situation Calculus

Historically, the first and most influential logical representation for causal reasoning was the *situation calculus* [McCarthy and Hayes, 1981]. In this representation, the problem solving domain is in one of a set of instantaneous states, or ***situations***. These situations are first-order objects. A ***fluent*** is a property of situations; intuitively, each fluent is either true or false in each situation. Traditionally, fluents have been represented as first-order predicates, e.g. "$\texttt{raining}(\texttt{s}_0)$" represents that the fluent $\texttt{raining}$ is true of the situation $\texttt{s}_0$.

We will often wish to attribute properties to objects in situations. For example. we might like to say the "block b is colored **green** in situation s_0". Ideally. we would define a function on domain objects which returns a fluent which is then applied to a situation. e.g. (**green**(b))(s_0). This is not possible when fluents are predicates. since predicate functions are not allowed in first-order logic. However. it is expressively equivalent to define predicates with domain objects as arguments in addition to the situation argument. as in **green**(b. s_0).

2.1.1 Actions and causal rules

In the situation calculus, an *action* is a function that maps a situation to a new situation. Rather than represent actions as actual first-order functions. they are usually *reified* using a "do" or "`result`" predicate. e.g. do(**shoot-gun**. s_0). The changes that an action makes are reflected by the differences in the values of the fluents between the original and resulting situations. This relationship between fluents and actions is captured using *causal rules*. based on material implication [Hayes. 1971] as in the following:

$$\texttt{clear}(b, s) \rightarrow \textbf{grasping}(b, \text{do}(\textbf{pickup}(b), s)) \tag{2.1}$$

In words. if the block b is clear in situation s, then the hand will be grasping b in the situation resulting from the application of action **pickup**(b) to s. Fluents that appear in the antecedent of a causal rule for an action are called *preconditions* of that action. and fluents that appear in the consequent are called *effects* of that action.

Two famous problems arose from the development of rules such as this: the *frame problem*. which asks how to succinctly represent what fluents do *not* change when an action is applied (the subject of Chapter 3), and the *qualification problem*, which asks how to keep the complexity of the antecedent at a reasonable level (the subject of Chapter 4).

2.1.2 Reified fluents

It has become more common to represent fluents as reified objects rather than predicates, just as actions have usually been represented as reified objects rather than functions. This approach uses a single "`holds`" [Lifschitz, 1987] predicate to capture the truth of a fluent with respect to a situation, e.g. `holds(raining`, s_0). There are two advantages to the reified approach: 1) fluent objects can be arguments to predicates such as `causes(raining,hair-wet)`, and 2) fluent objects can be the value of a function on domain objects, such as `holds(on(block,table)`, s_0). Using predicates, this would be represented in the slightly less intuitive way on(b, `table`, s_0) [Hayes, 1971, Bacchus *et al.*, 1989].

Advantage 1 above allows causal rules to be expressed as relations rather than explicit implications. Lifschitz [1987] used two predicates, "**causes**" which captures when an action causes a fluent to be true, and "**precond**" which captures when a fluent is a precondition of the successful performance of an action (i.e. the members of the antecedent in (2.1)). For example, (2.1) would instead be written as:

$$\texttt{precond(clear(}b\texttt{),pickup(}b\texttt{))}$$

$$\texttt{causes(pickup(}b\texttt{),grasping(}b\texttt{))}$$

These predicates are given meaning via the following *Law of Change*:

$$\texttt{causes}(a,\epsilon) \wedge \forall p[\texttt{precond}(p,a) \rightarrow \texttt{holds}(p,s)] \rightarrow \texttt{holds}(\epsilon,\texttt{do}(a,s))$$

In words, if all of action a's preconditions are true, and a causes fluent ϵ to be true, then ϵ will be true when a is applied. With this approach, it is important to have axioms capturing what is not a precondition of an action: Lifschitz achieved this by minimizing the precond predicate (cf. Section 4.4). Negated preconditions and effects can also be represented if a function is defined that negates fluents, e.g.

$$\texttt{holds(not(}f\texttt{)},s) \equiv \neg\texttt{holds}(f,s)$$

allowing assertions like **causes(putdown**(b)**,not(grasping**(b)**))**.

2.2 Explicit Time Lines

The situation calculus provides a strange model of time, since a new situation is created only by the application of an action. This makes it hard to represent facts such as "the bank will be open at 10 AM", since there is no global temporal reference with which to define "10 AM" or relative times like "an hour from now". This deficiency is addressed by adding an *explicit time line* to the representation. The result looks surprisingly like the situation calculus, but I will point out some subtly important differences in the meaning of *actions*.

2.2.1 Moments

Instead of situations, an explicit temporal representation uses a totally ordered set of *moments* (a.k.a. time points). A *successor function* for this set maps a moment to that unique moment which comes directly afterwards. Such a function exists if and only if the set of moments is countable; if this is so then the temporal model is said to be *discrete*. Otherwise, the temporal model is said to be *continuous*. Although the psychological, philosophical and computational ramifications of this choice are very interesting [McDermott, 1982], they are tangential to the main points of this thesis.

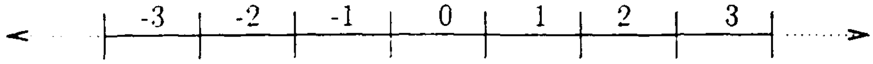


Figure 2.1: A discrete, linear, unbounded time line of moments

For simplicity, I will concentrate on discrete representations and therefore assume the existence of a successor function.

I will use integer constants to stand for moment constants, in order to easily convey the ordering of moments in examples. Integers are a reasonable choice since by definition, any countable, totally ordered set is isomorphic with a subset of the integers. I will also employ operations on integers as operations on moments. In particular, the successor of moment m is $m + 1$, or equivalently, m'. Intuitively, moments partition a (non-branching) time line into pieces of equal size each labeled with an integer, as shown in Figure 2.1.

2.2.2 Fluents

Fluents are either true or not of moments, just as they were for situations. One of the predicates "**holds**" [Georgeff, 1987, Weber, 1988] or "T" [Shoham, 1988] is usually used to relate reified fluents to moments, as in **holds(raining**, 15). Loui [1987b] presented a variation using an "@" predicate in an infix form, e.g. **raining**@15. Shoham [1987] even defined the syntax and semantics of a temporal logic as a substitute for first-order, although Bacchus *et al.* [1989] showed that the resulting logic was no more expressive than adding an additional temporal argument to fluent-predicates, e.g. clear(b, 22).

2.2.3 Actions

Since a moment's successor is defined by the successor function and not "produced" by the application of an action, the semantics of action is profoundly changed. Georgeff [1987] proposed that actions are now ***relations*** between moments instead of functions; thus an action constrains what fluents can be true on future moments. For example, the following sentence defines the successful performance of a "**pickup**" action:

$$\mathbf{occurs}(\mathbf{pickup}(b), m, m+\delta) \equiv (\neg\mathbf{holds}(\mathbf{grasping}(b), m) \wedge \mathbf{holds}(\mathbf{grasping}(b), m+\delta))$$

where the "**occurs**" predicate applies the reified action relation **pickup**(b) to two moments separated by the duration δ. Since δ is usually assumed to be one, i.e. actions constrain moments and their immediate successors, it is notationally sufficient to drop the third argument and simply write **occurs**(**pickup**(b), m) [Georgeff, 1987, Weber, 1988].

2.2.4 Causal rules

For an explicit time line, a causal rule is a material implication that has the truth of fluent preconditions and the occurrence of an action imply the truth of a fluent during the next moment, i.e.

$$\texttt{holds}(\texttt{clear}(b), m) \land \texttt{occurs}(\texttt{try-pickup}(b), m) \rightarrow \texttt{holds}(\texttt{grasping}(b), m') \quad (2.2)$$

Intuitively, the action $\texttt{try-pickup}$ is an *attempt* at picking up the block, which will succeed if the block is clear. Some approaches, rather than represent (2.2) explicitly, split it into two axioms like the following:

$$\texttt{holds}(\texttt{clear}(b), m) \land \texttt{occurs}(\texttt{try-pickup}(b), m) \rightarrow \texttt{occurs}(\texttt{pickup}(b), m)$$

$$\texttt{occurs}(\texttt{pickup}(b), m) \rightarrow \texttt{holds}(\texttt{grasping}(b), m')$$

The first axiom expresses a *generation* [Goldman, 1970] relation between the two actions $\texttt{try-pickup}(b)$ and $\texttt{pickup}(b)$, i.e. that in all contexts such that $\texttt{holds}(\texttt{clear}(b), m)$, the $\texttt{try-pickup}(b)$ constitutes a successful $\texttt{pickup}(b)$ action. The second axiom expresses the effects that an action like $\texttt{pickup}(b)$ is known to have. When taken together, these two axioms imply (2.2). These two approaches to causal rules are for the most part interchangeable.

2.2.5 Simultaneous actions

It is simple to represent simultaneous actions when using an explicit time line: if both actions $\texttt{a}$ and $\texttt{b}$ occur at moment $\texttt{m}$, then both the literals $\texttt{occurs(a,m)}$ and $\texttt{occurs(b,m)}$ will be true. The ability to represent simultaneous action has often been cited as a motivation for explicit time over the situation calculus [Georgeff, 1987, Morgenstern and Stein, 1988, Weber, 1988], but in actuality, the situation calculus can represent simultaneous action if we add action *types* [Schubert, 1989], using literals of the form $\texttt{isa}(a, A)$ where action a is of type A. We can define the effects when an action of type $\texttt{Pickup}(b)$ is applied (as opposed to the action *instance* $\texttt{pickup}(b)$ used above) in the following way:

$$\texttt{clear}(b, s) \land \texttt{isa}(a, \texttt{Pickup}(b)) \rightarrow \texttt{grasping}(b, \texttt{do}(a, s))$$

That is, whenever an action of type $\texttt{Pickup}(b)$ is applied to a clear block b, the robot will be grasping b. Since actions may be of multiple types, the following is a theorem:

$$\begin{gathered}\texttt{clear}(\texttt{b}_1, s) \land \texttt{clear}(\texttt{b}_2, s) \land \texttt{isa}(a, \texttt{Pickup}(\texttt{b}_1)) \land \texttt{isa}(a, \texttt{Pickup}(\texttt{b}_2)) \rightarrow \\ \texttt{grasping}(\texttt{b}_1, \texttt{do}(a, s)) \land \texttt{grasping}(\texttt{b}_2, \texttt{do}(a, s))\end{gathered}$$

which captures the simultaneous application of two action types $\texttt{Pickup}(\texttt{b}_1)$ and $\texttt{Pickup}(\texttt{b}_2)$. Note that this discussion, and indeed the entire thesis, ignores the interactions of simultaneous actions, e.g. trying to pick up two blocks at the same time with a single arm[1]. For enlightening discussion about this problem see Georgeff [1987] or Pelavin [1986, 1988].

The concept of simultaneous action has caused much confusion in comparisons between the situation calculus and explicit time representations. This is because, as demonstrated above, they involve two different notions of what an action is. In the situation calculus, an action is a complete specification of the changes that happen from one situation to the next. With explicit time, an action is a partial specification of the changes from one moment to the next (this distinction will be important in Chapter 3, when I discuss solutions to the frame problem). Above we saw that the situation calculus can represent partial specifications through the addition of action types. Conversely, an explicit temporal model can assert a complete specification of the changes as in the following way:

$$\mathtt{occurs}(a, 4) \equiv (a = \mathtt{pickup(block1)})$$

This axiom says that pickup(block1) is the *only* action to occur at time 4. Thus situation calculus and explicit time models are not as expressively different with respect to simultaneous action as is typically claimed. However, it should be clear that explicit time models are more convenient for expressing partial knowledge about action occurrences. Since this will be the case for agents in realistic domains, explicit time is more appropriate than the situation calculus as a representation for causal reasoning.

2.3 Temporal Intervals

Allen [1984] suggested a generalization of the above temporal model where temporal objects are *intervals*, which may overlap, as opposed to moments, which may not. In this approach, the stretches of time over which fluents are asserted to hold need not conform to some *a priori* partitioning of the time line, producing a more elegant representation of fluents over time as shown in Figure 2.2. Intervals have been influential in temporal representation, but it is interesting to note that many uses of intervals have actually defined them in terms of their end-moments [Shoham, 1988, Williams, 1986]. Conversely, moments can be defined in terms of the intersection of two intervals!

There has been significant discussion about the relative merits of intervals versus moments (a.k.a. points), but in fact there have been no convincing arguments demonstrating expressive or computational advantages of either approach. Therefore, this thesis will concentrate on representations using moments, to facilitate comparisons with the most relevant work in causal reasoning.

[1] Actually, the statistical approach that starts with Chapter 5 offers a general mechanism for reasoning about interactions, since conditioning on more information (the occurrence of additional actions) can change the probability of a prediction.

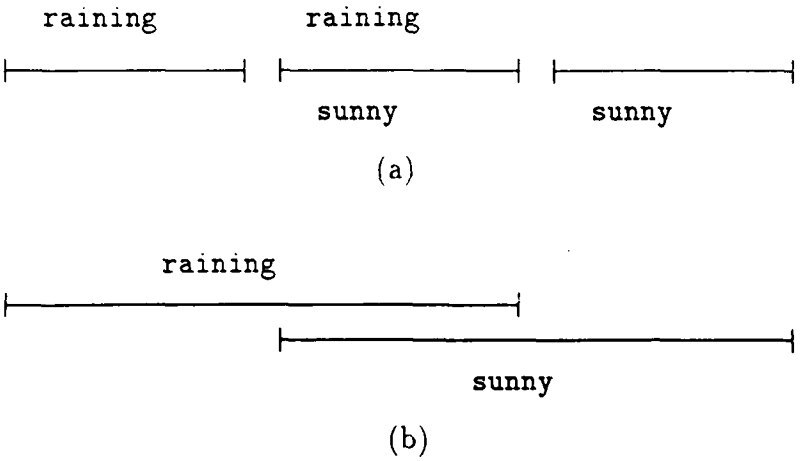


Figure 2.2: Comparative temporal representations using moments and intervals: (a) uses disjoint moments and (b) uses overlapping intervals.

2.4 Fluents as Sets

A main point of this chapter is that the notational differences in causal representations are mostly superficial. However, the choice of representation does affect the simplicity and clarity of solutions to problems they are used to solve. For example, reified fluent representations are more convenient than non-reified versions when applying circumscription [McCarthy, 1984, Lifschitz, 1987], since predicates on fluents are easy to construct. Also, Shoham's special semantics for time facilitates his operation of chronological minimization [Shoham, 1988]. Therefore, when presenting the work of others, this thesis will use the original notation or some variant that I believe will make the contribution clearer.

This thesis also uses a superficially different notation for the purposes of simplicity and clarity. Instead of representing fluents as atomic objects which are related to moments by a "truth" predicate, I will represent fluents as explicit *sets of moments* during which the fluent is "true". This representation is motivated by the work on causal statistics in Chapter 5, since the basic mechanisms I will use for statistical inference are based on set-theoretic concepts, e.g. relative frequencies of common elements of sets. The notation is essentially a discrete explicit temporal model, as described previously in this chapter. In addition to the syntactic representation of fluents as sets, a notable difference is that actions are a special case of fluents. These differences are described in the following sections.

2.4.1 Moments and fluents

A fluent is simply a set of moments. Intuitively, a fluent is "true" over a moment if and only if the moment is an element of the fluent. Thus fluents and moments are related by set membership, where "15 $\in$ `raining`" means that it was raining at moment 15. This is essentially the same relation as T(`raining`, 15) from reified explicit temporal models [Shoham, 1988], although casting fluents as sets has the following advantage: set theory provides operations for manipulating fluents, whereas those operations must be explicitly defined in the standard notation. For example, consider representing the logical relationship between a fluent and its negation. In the standard reified approach, a new function must be defined, e.g. $\mathrm{T}(\mathtt{not}(f), m) \equiv \neg\mathrm{T}(f, m)$. This function already exists in set theory as the complement operator, i.e. the set of moments that a fluent does not hold is equal to the complement of the set of moments that the fluent does hold ($\neg(m \in f) \equiv m \in \overline{f}$). The set operations of intersection and union can also be used for logical "and" and "or" of fluents, i.e.

$$(m \in f \wedge m \in g) \equiv (m \in f \cap g)$$

$$(m \in f \vee m \in g) \equiv (m \in f \cup g)$$

Following this course, we could also write sentences that describe the material implication of fluent memberships, i.e.

$$\forall m[m \in \mathtt{raining} \rightarrow m \in \mathtt{precipitating}]$$

as the (improper) subset relation, i.e.

$$\mathtt{raining} \subseteq \mathtt{precipitating}$$

Despite the comparative succinctness of the latter sentence, I will use the material implication version because it is more recognizable.

2.4.2 Temporally-relative fluents, actions, and causal rules

The representation of fluents as sets is a more general concept than the intuitive reading of a fluent as the set of moments a particular "physical" domain property is true. For example, we could define a fluent to be "$\{m : m' \in \mathtt{raining}\}$", i.e. the set of moments such that it will be raining during the *next* moment. Similarly, a fluent can be defined to be another fluent two moments ahead, four moments prior, etc. I call fluents defined in this way ***temporally-relative***. This construction will be particularly useful in Chapter 5, where I use statistics like "the proportion of moments whose successors are in the fluent `raining`, given that those moments are in the fluents `cloudy` and `humid`".

Sets of moments can also capture changes in other fluents. For example, we could define a fluent pickup-b in the following way:

$$m \in \texttt{pickup-b} \equiv (m \in \overline{\texttt{grasping-b}} \wedge m' \in \texttt{grasping-b}) \qquad (2.3)$$

Informally, block b is picked up at moment m if and only if b is not grasped at m but is at m'. This manner of constructing fluents subsumes the role of *actions* in explicit temporal approaches, making the addition of distinct action objects superfluous. Therefore, in this notation, actions are treated as a special kind of fluent defined in terms of other fluents over multiple moments. This approach has the practical advantage of not requiring a special syntax for the construction and application of actions.

Representing actions as fluents also has philosophical advantages. Typically, causal reasoning systems have had no mechanism for inferring the occurrence of actions other than that the reasoning agent "wills" them to occur. This is fine in the rare domains that permit the *single agent assumption*, namely that all changes in the domain are directly caused by the reasoning agent. However, when so-called *external events* (not caused by the reasoning agent) are allowed, the problem arises of how to deduce that they occurred, and therefore predict changes based on their occurrence. Some approaches [Morgenstern and Stein, 1988, Georgeff, 1987, Pelavin, 1988] allow *generation* [Goldman, 1970] rules which imply the occurrence of arbitrary events based on the occurrence of *other* events, but at some level external event occurrence should be deduced *purely in terms of observable fluents*. Therefore, external actions must ultimately be expressible as a combination of fluents, which in my notation is just another fluent.

It is also reasonable to treat "internal" events, i.e. an agent's actions, as a complex fluent. When an agent "wills" an action, the agent causes primitive fluents to hold over the moments involved. For example, when an agent opens a door, she causes a torque to be applied to the door handle on one moment and then a force vector to be applied to the edge of the door during the next moment.

2.4.3 Domain objects and fluents

An example in the previous subsection used a fluent pickup-b which applied to a *particular* block b. Thus distinct fluents would need to be defined for each block that could be picked up. It will be more convenient to be able to define fluents that apply to a class of domain objects, say any block. Recall that with the reified fluent approaches described in the beginning of this chapter, this was done through the definition of functions that return fluents, e.g. pickup(b) would return the fluent corresponding to pickup-b. In my notation, this is done using sets of tuples which contain moments. For example, to generalize (2.3) for all blocks b, I would write:

$$\langle m, b\rangle \in \texttt{pickup} \equiv (\langle m, b\rangle \in \overline{\texttt{grasping}} \wedge \langle m', b\rangle \in \texttt{grasping})$$

where `pickup` and `grasping` are block-fluents, i.e. subsets of `blocks` × `moments`. Again, this notation will be more convenient when representing statistics later in this thesis. To minimize confusion, most of the examples in this thesis will use simple fluents that are not generalized over domain objects.

2.5 Chapter Summary

Logical causal representations differ not so much in expressibility, but the ease with which we can encode various solutions to causal reasoning problems. I briefly described several popular representations, including the situation calculus, explicit time and action occurrence, and interval-based time. In addition, I presented a superficially different representation, based on the representation and manipulation of fluents as sets, which will be more convenient to use in later chapters when discussing statistical causal inference.

Chapter 3

The Frame Problem and Domain Dependence

The *frame problem*, so named for the part of a cartoon that remains the same when an object moves, proclaims the need for a mechanism that decides what world properties do not change during an action as well as deciding what does change. It was originally presented as a technical difficulty within the situation calculus but in the intervening years the frame problem has surfaced in all formal approaches to reasoning about action and causality. Judging by the amount of research and discussion it has inspired, it is generally regarded as THE problem of causal reasoning. To establish a context for my approach, in this section I review the more significant progress on the problem.

3.1 Frame axioms

In the situation calculus, actions are objects that may be applied to a world state to produce a new state. The fact that an action (such as **pickup(a)**) produces a world state in which a property (such as **grasping(a)**) holds is typically represented as:

$$\forall s[\texttt{holds(grasping(a)}, \texttt{result(pickup(a)}, s))] \tag{3.1}$$

Formulas such as these very concisely capture what does change, but we still must have a mechanism for capturing what does not change: we should be able to deduce that picking up **a** does not change the position of **b**. One way to provide the inferential power is to encode explicit *frame axioms*, as in the following:

$$\forall s[\texttt{holds(on(b,table)}, s) \equiv \texttt{holds(on(b,table)}, \texttt{result(pickup(a)}, s))] \tag{3.2}$$

Axioms such as these, for all combinations of actions and properties, would give us the ability to exactly determine the effects of actions.

This approach is rarely taken seriously because of the large number of such frame axioms needed for a complex domain. For a domain with n properties and m actions, we would require $n \times m$ frame axioms in the worst case. Even when one considers the use of quantification over objects, the number of frame axioms needed is still very daunting. Faced with this obstacle, work on the frame problem turned to the search for a more economical domain-independent mechanism that does the work of the frame axioms. It remains the dominant direction to this day, although refuting its fundamental premises is an important part of this thesis (cf. Section 3.5).

It is important to note a subtle fallacy in the above historical argument against frame axioms. The argument interprets a large number of theorems as a space requirement, yet there could be a compact implementation representation that together with an inference procedure generates such a large number of theorems. In fact, another formulation of the frame axiom (that was only recently made explicit) is as inferentially powerful yet suggests a more economical implementation. I will discuss this formulation in Section 3.7; first I have a bit more history to elaborate.

3.2 Minimizing changes

McCarthy [1984] suggested that circumscription could be applied to solve the frame problem to formulate that properties should remain static by default. His solution was to introduce a predicate like the following:

$$\texttt{abnormal}(a,p,s) \equiv \neg(\texttt{holds}(p,s) \equiv \texttt{holds}(p,\texttt{result}(a,s))) \qquad (3.3)$$

This predicate captures when properties change due to applied actions. For example, if $\neg\texttt{holds}(\texttt{Grasping}(\texttt{A}),\texttt{S}_0)$ then from (3.1) and (3.3) we can infer:

$$\texttt{abnormal}(\texttt{Pickup}(\texttt{A}),\texttt{Grasping}(\texttt{A}),\texttt{S}_0)$$

The nonmonotonic component is that we now circumscribe the **abnormal** predicate. If we assume our domain theory cannot prove $\texttt{abnormal}(\texttt{Pickup}(\texttt{A}),\texttt{On}(\texttt{B},\texttt{Table}),\texttt{S}_0)$, then its negation will become a theorem after circumscription[1] which together with (3.3) entails the frame axiom in (3.2). I call this approach to the frame problem that of *minimizing changes*.

3.2.1 The shooting problem

Hanks and McDermott [1986] showed that the minimizing changes approach exhibits an unnatural temporal ambiguity, demonstrated by the "Yale Shooting Problem". The standard exposition involves a planning agent who loads a gun, waits momentarily, and then fires the gun at a would-be victim. They used a situation calculus representation of the scenario, which I simplify by removing the gun-loading action leaving:

$$\texttt{holds}(\texttt{loaded},\texttt{s}_0) \wedge \texttt{holds}(\texttt{alive},\texttt{s}_0)$$

[1] Ignoring the possibility of multiple minimal models, of course.

$$\texttt{holds(loaded},s) \rightarrow \neg\texttt{holds(alive,result(shoot},s))$$

$$\neg\texttt{holds(loaded,result(unload},s)) \tag{3.4}$$

If $\mathtt{s}_1 = \texttt{result(wait,}\mathtt{s}_0)$ and $\mathtt{s}_2 = \texttt{result(shoot,}\mathtt{s}_1)$, the theory has many different models (i.e. interpretations of the formulas), including the following two classes:

$$\begin{array}{cc}
\mathcal{A} & \mathcal{B} \\
\texttt{holds(loaded,}\mathtt{s}_1) & \neg\texttt{holds(loaded,}\mathtt{s}_1) \\
\neg\texttt{holds(alive,}\mathtt{s}_2) & \texttt{holds(alive,}\mathtt{s}_2) \\
\texttt{abnormal(shoot,alive,}\mathtt{s}_1) & \texttt{abnormal(wait,loaded,}\mathtt{s}_0)
\end{array} \tag{3.5}$$

In $\mathcal{A}$, alive is abnormal with respect to shoot, and in $\mathcal{B}$ loaded is abnormal with respect to wait. The popular intuition is that models of class $\mathcal{A}$ are reasonable to consider, but class $\mathcal{B}$ models are not, since in $\mathcal{B}$ the gun is unloaded during the wait without explanation. Both classes, however, are minimal with respect to the abnormality predicate defined above. Hanks and McDermott interpreted this to mean that the minimizing changes approach does not produce intuitively correct inferences, and that some other solution to the frame problem is needed that makes the distinction between $\mathcal{A}$ models and $\mathcal{B}$ models.

3.2.2 Chronological minimization

Almost immediately, three researchers [Shoham, 1986, Lifschitz, 1986, Kautz, 1986] responded to the shooting problem by developing new circumscription variants that resolved the ambiguity. The approach they used was essentially the same, now called *chronological minimization* (Shoham's term). It stems from the observation that in (3.5), the abnormality tradeoff involves two different states, $\mathtt{s}_0$ and $\mathtt{s}_1$. They suggested that the distinction Hanks and McDermott called for is that $\mathtt{s}_1$ is derived from $\mathtt{s}_0$ and not vice versa, so in a sense $\mathtt{s}_1$ is chronologically after $\mathtt{s}_0$. If we augment our circumscription to prefer chronologically later abnormality when given a choice, $\mathcal{A}$ models will be chosen over $\mathcal{B}$ models. Since this involves the notion of temporal order, these approaches used explicit temporal representations.

As popular as the chronological minimization technique was when it was introduced, it is now generally considered a dead end. It does give the results Hanks and McDermott proposed for the shooting problem, but it produces other results that question its fundamental soundness. The first such problem was mentioned by Kautz himself:

> Recall the example using persistence to infer that my car is in the parking lot. Suppose I learn at time 1000 that my car is gone. Using the techniques just described I can infer that the car was in the parking lot up to the shortest possible time before I knew it was gone. This is clearly an unreasonable inference. Someone could have stolen it five minutes after I

> left it there: I have no reason to prefer an explanation in which it vanished five seconds before I glanced out my office window. [Kautz, 1986]

This problem was later rediscovered by Haugh [Haugh, 1987] and then dubbed the problem of *unexpected results* by Goodwin [Goodwin, 1988]. Another problem mentioned by Haugh is even worse, which I summarize with the following prose example: suppose I tell you two previously unknown facts that either a) the president died yesterday, or b) Columbus discovered America in 1492. Chronological minimization tells you that *a* should default to true and *b* to false merely because of temporal order.

3.3 Circumscribing Causality

Lifschitz [Lifschitz, 1987] and Haugh [Haugh, 1987] independently pursued an approach to the shooting problem that does not use chronological minimization. I will review Lifschitz's version because it is slightly simpler. In Chapter 2, I discussed Lifschitz's use of the **causes** and **precond** predicates in the situation calculus with reified fluents. Instead of using the universally quantified formulas in (3.4), he represented the shooting problem as[2]:

$$\begin{array}{c}\mathtt{causes(unload,not(loaded))}\\ \mathtt{causes(shoot,not(alive))}\\ \mathtt{precond(loaded,shoot)}\end{array} \qquad (3.6)$$

Recall that these literals were related to changes in fluents by the *Law of Change*:

$$\mathtt{causes}(a,e) \land \forall p[\mathtt{precond}(p,a) \rightarrow \mathtt{holds}(p,s)] \rightarrow \mathtt{holds}(e,\mathtt{do}(a,s)) \qquad (3.7)$$

To solve the frame problem, Lifschitz added an axiom that guarantees that every change must be due to an appropriate action with its preconditions satisfied. He called this axiom the *Law of Inertia*:

$$\begin{array}{c}(\neg\mathtt{causes}(a,\mathtt{not}(p)) \land \neg\mathtt{causes}(a,p)) \lor \exists q[\mathtt{precond}(q) \land \neg\mathtt{holds}(q,s)] \rightarrow\\ (\mathtt{holds}(p,s) \equiv \mathtt{holds}(p,\mathtt{do}(a,s))))]\end{array} \qquad (3.8)$$

In words, the value of a fluent *p* will stay the same after an action *a* is applied if *a* does not cause *p* or **not**(*p*), *or* some precondition of *a* does not hold. If the **causes** predicate is minimized (i.e. the negations of all **causes** literals not in (3.6) added), the Law of Inertia derives what properties do not change when an action is applied. In particular, since **wait** does not appear in any of the **causes** assertions of (3.6), no property will change when a **wait** action is applied, including **loaded**. Thus class $\mathcal{B}$ models are excluded, solving the shooting problem.

[2] Actually, Lifschitz had a third argument in his **causes** literals, namely the truth value that the fluent will have in the resulting state. In my description, that argument is unnecessary due to the definition of the **not**() function for fluent negation. This technique also simplifies his laws of change and inertia to follow.

	CARRY(?OBJECT,?LOCATION1,?LOCATION2)
Pre:	AT(ME,?LOCATION1), GRASPING(?OBJECT)
Del:	AT(ME,?LOCATION1), AT(?OBJECT,?LOCATION1)
Add:	AT(ME,?LOCATION2), AT(?OBJECT,?LOCATION2)

Figure 3.1: STRIPS action for the agent carrying an object

3.3.1 The STRIPS Assumption

Lifschitz' solution to the frame problem similar to the *STRIPS Assumption*, a.k.a the *sleeping dog strategy*. The STRIPS planner [Fikes and Nilsson, 1971] introduced a very influential representation of action using precondition, delete, and add lists, an example of which appears in Figure 3.1. The current state of the domain is a database of facts like AT(BLOCK5,LOC12), and when an action is applied whose preconditions are true (according to the database), those facts on the delete list are removed and those on the add list inserted. The STRIPS Assumption summarizes the inherent solution to the frame problem: when an action is applied, database facts that are not explicitly on the action's delete list are not affected. This technique is identical in principle to Lifschitz' use of nonmonotonicity, where "p is not on a's delete list" translates to $\neg$causes(a, p).

3.3.2 Why minimizing causes cheats the frame problem

Lifschitz's solution to the shooting problem, and therefore the frame problem, relies on two crucial assumptions: 1) that the reasoning agent has complete knowledge of what fluents are affected by the application of each action, and 2) that the reasoning agent has complete knowledge about which actions are applied. Given these two assumptions, the frame problem is easy to solve. Assumption 1 is implemented explicitly by minimizing the **causes** predicate, and merely suggests that the agent is an expert on the domain. This is perhaps too strong, but not entirely unrealistic. Assumption 2 is much more subtle, being implicit in the use of action *constants* in the do function. Recall the discussion in Section 2.2.5, which demonstrated that although it is straightforward to represent simultaneous actions in the situation calculus, one must resort to action types to represent partial knowledge about which actions occurred. Since Lifschitz uses action constants such as loaded and wait, he is invoking the assumption that the agent knows precisely which actions occur, and more importantly for the frame problem, which actions do *not* occur.

The "omniscience" assumption is implicit in virtually every use of the situation calculus. Unfortunately, it is doubtful that many useful domains exist in which this assumption is justified.

3.4 The Persistence Problem

The incarnation of the frame problem in causal representations involving an explicit time line is often called the *persistence problem.*[3] Recently, some researchers have attempted to solve the persistence problem by importing Lifschitz's situation calculus solution to the frame problem [Lifschitz, 1987]. Surprisingly, the pursuit of such a formal system has discovered that the generalization is quite tricky.

The difficulty arises from the new lack of complete knowledge about which actions actually occur at a given time. Lifschitz's solution to the frame problem depends on knowing the action *constant* that produces a new state. No matter how complex this action may be, its functional character tells us exactly what occurred. Thus in the shooting problem, an `unload` action did not occur precisely because a `wait` action *did.* Whereas do(a, s) is many-to-one, occurs(a, s) is many-to-many.

3.4.1 Minimizing action occurrences

A significantly new solution to the frame problem is required when considering explicit time and action occurrences. Several researchers [Morgenstern and Stein, 1988, Sandewall, 1988, Weber, 1988] have suggested circumscribing action occurrences in addition to causal connections. I will concentrate on my own previous work on the subject since its notation fits better into this discussion, and I feel justified in making certain simplifications.

An analogue to the Law of Inertia for explicit occurrence can be established by the following *Axiom of Blame*:

$$\begin{gathered}(\mathtt{holds}(p,t) \wedge \neg\mathtt{holds}(p,t')) \rightarrow \\ \exists a[\mathtt{occurs}(a,t) \wedge \forall u[\mathtt{occurs}(a,u) \rightarrow \neg\mathtt{holds}(p,u')]]\end{gathered} \tag{3.9}$$

A predicate defined by $\lambda a.p[\mathtt{occurs}(a,u) \rightarrow \neg\mathtt{holds}(p,u')]$ is circumscribed to minimize causal connections. Action occurrences are minimized by simply circumscribing the `occurs` predicate.

Now let us see how the shooting problem is solved by this approach. We will use the following version of the shooting scenario with explicit time and action occurrence:

$$\mathtt{holds}(\mathtt{loaded},0) \wedge \mathtt{holds}(\mathtt{alive},0)$$

$$\mathtt{occurs}(\mathtt{wait},0) \wedge \mathtt{occurs}(\mathtt{shoot},1)$$

$$\forall m[\mathtt{holds}(\mathtt{loaded},m) \wedge \mathtt{occurs}(\mathtt{shoot},m) \rightarrow \neg\mathtt{holds}(\mathtt{alive},m')]$$

$$\forall t[\mathtt{occurs}(\mathtt{unload},m) \rightarrow \neg\mathtt{holds}(\mathtt{loaded},m')]$$

[3] Normally I wouldn't support a new term for the same old problem, but the word "persistence" is so much more suggestive than "frame".

Before circumscribing, this formulation has many different models including the following three classes:

$\mathcal{A}$	$\mathcal{B}$	$\mathcal{C}$
$\texttt{occurs}(a, 0) \rightarrow a = \texttt{wait}$	$\texttt{occurs}(a, 0) \rightarrow a = \texttt{wait}$	$\texttt{occurs}(\texttt{unload}, 0)$
$\texttt{holds}(\texttt{loaded}, 1)$	$\neg\texttt{holds}(\texttt{loaded}, 1)$	$\neg\texttt{holds}(\texttt{loaded}, 1)$
$\neg\texttt{holds}(\texttt{alive}, 2)$	$\texttt{holds}(\texttt{alive}, 2)$	$\texttt{holds}(\texttt{alive}, 2)$

As before, the desired models fall into class $\mathcal{A}$. Class $\mathcal{B}$ models are eliminated when we circumscribe the causal connections, since we know that only a **wait** occurs at 0 and we can make the default that **wait** does not affect loaded. Class $\mathcal{C}$ models are not eliminated until we circumscribe occurs, thus creating the default $\neg\texttt{occurs}(\texttt{unload}, 0)$. This leaves only the intuitively correct models of class $\mathcal{A}$. Incidentally, it is not necessary to check that unnamed action occurrences are circumscribed away, since by circumscribing causal connections those actions have no effects.

3.4.2 The generation/minimization problem

Unfortunately, although the "minimizing occurrences" approach solves the shooting problem in the popular manner, it exhibits some serious technical and philosophical problems. One problem appears when we try to represent action negation, as in the following:

$$\texttt{occurs}(\texttt{HoldDoorOpen}, 0) \equiv \neg\texttt{occurs}(\texttt{LetDoorClose}, 0)$$

One may wish actions like **HoldDoorOpen**; after all, the agent applies a force in order to achieve an effect. When we circumscribe, we cannot assume that neither action occurred, thus creating an ambiguity. It may be reasonable to live with the model ambiguity that these actions instigate, but this ambiguity did not exist until we simply added another action object to our ontology.

An even more serious problem appears when we try to represent the generation relation. In their formal theory that circumscribes occurrences, Morgenstern and Stein explicitly allow the occurrence of actions in certain contexts to imply the occurrence of other actions, in order to "express causal chains of action" [Morgenstern and Stein, 1988], i.e. generation. For example, when representing the shooting problem we may wish to use an axiom of the form:

$$\texttt{occurs}(\texttt{pulltrigger}, m) \wedge \texttt{holds}(\texttt{loaded}, m) \rightarrow \texttt{occurs}(\texttt{murder}, m) \qquad (3.10)$$

Note what happens in the following scenario: at some moment we know that a **pulltrigger** occurs, but we do not know whether **loaded** holds. When we circumscribe **occurs**, we infer that a **murder** does not occur. This fact taken with (3.10) implies that loaded does not hold. This is clearly a ridiculous consequence, especially considering that the description of the problem said we do not know whether the gun is loaded. I call this the *generation-minimization problem*. In fact, this problem applies to more than just representing generation: it appears whenever a system has a mechanism for inferring the occurrence of an action under some conditions. Action minimization will arbitrarily decide that those conditions are false when the action doesn't necessarily occur.

These two problems are fixed somewhat by being careful about which actions' occurrences are circumscribed [Weber, 1988], but the result is not very appealing. The distinction between "basic" and "inferred" actions is reminiscent of the artificial distinction in STRIPS between essential and inessential database facts. In both cases the distinction is more a property of the way knowledge is represented than of the domain.

Another objection stems from the observation that minimizing occurrences uses nonmonotonicity in quite a different way than minimizing causal connections. Like the original minimizing changes approach, minimizing occurrences makes a claim about what actually is happening in the domain at a specific time, rather than making a claim about the domain theory. Morgenstern and Stein [1988] say that minimizing occurrences 'formalizes the intuition that we typically reason that events in a chronicle happen only when they have to happen'. Certainly this intuition is popular, but is it *reasonable*? In the next section, I discuss this issue with respect to the entire history of work on the frame problem.

3.5 The Myth of Domain Independent Persistence

The challenge of the shooting example is to supply a domain-independent mechanism which produces the intuitive conclusion that the gun stays loaded during a wait. Many published responses to this challenge have been subsequently refuted here and elsewhere [Haugh, 1987, Lifschitz, 1987, Loui, 1987a] on the grounds that they produce unintuitive results in some cases. This is at least partially due to the fact that the mandate for domain-independent mechanisms is ill-founded. All discussion that I have seen upholds Hanks and McDermott's moral that a more powerful *domain-independent* persistence mechanism is needed. If this is true, then a scenario with the same morphology (i.e. only changing constants) should have the same intuitive inferences. I refute this with the following example, called the *hesitation scenario*:

> A famous desert bird is enjoying a bit of seed while standing on a large black "X". Overhead, an anvil is suspended over a pulley, with the rope leading to an equally famous desert canine hiding behind a rock. The would-be villain knows that if he lets loose of the rope now, he will finally get that blasted bird. However, he suddenly notices he must wait first, as the rope is tangled around his foot. What happens to the bird?

A simple representation of this scenario can be achieved by giving new meanings to the predicates in the shooting scenario above, which I signify by new predicate names in the following version:

$$\texttt{holds}(\texttt{onx}, 0) \wedge \texttt{holds}(\texttt{alive}, 0)$$

$$\texttt{occurs}(\texttt{wait}, 0) \wedge \texttt{occurs}(\texttt{drop}, 1)$$

$$\forall t[\texttt{holds}(\texttt{onx}, m) \wedge \texttt{occurs}(\texttt{drop}, m) \rightarrow \neg\texttt{holds}(\texttt{alive}, m')]$$

$$\forall t[\texttt{occurs}(\texttt{leave}, m) \rightarrow \neg\texttt{holds}(\texttt{onx}, m')]$$

$$\begin{array}{cc}
\mathcal{A} & \mathcal{C} \\
\neg\texttt{occurs}(\texttt{leave}, 0) & \texttt{occurs}(\texttt{leave}, 0) \\
\texttt{holds}(\texttt{onx}, 1) & \neg\texttt{holds}(\texttt{onx}, 1) \\
\neg\texttt{holds}(\texttt{alive}, 2) & \texttt{holds}(\texttt{alive}, 2)
\end{array}$$

Given that this bird is not known to linger, $\texttt{occurs}(\texttt{leave}, 0)$ seems more likely than $\neg\texttt{occurs}(\texttt{leave}, 0)$ and thus $\mathcal{C}$ seems more intuitive than $\mathcal{A}$. This choice directly opposes the popular interpretation of the shooting problem.

This point is demonstrated with a simpler example, where I replace the constant `wait` with `sleep`, as follows:

$$\texttt{holds}(\texttt{loaded}, 0) \wedge \texttt{holds}(\texttt{alive}, 0)$$

$$\texttt{occurs}(\texttt{sleep}, 0) \wedge \texttt{occurs}(\texttt{shoot}, 1)$$

$$\forall m[\texttt{holds}(\texttt{loaded}, m) \wedge \texttt{occurs}(\texttt{shoot}, m) \rightarrow \neg\texttt{holds}(\texttt{alive}, m')]$$

$$\forall t[\texttt{occurs}(\texttt{unload}, m) \rightarrow \neg\texttt{holds}(\texttt{loaded}, m')]$$

The intuitive scenario becomes: you point a loaded gun at a would-be victim, sleep for a few hours, then shoot. The representation of this version has the same syntactic structure, so a domain-independent treatment of the problem will be the same. Yet, the intuitive understanding of the problems are different: whereas we would expect the gun to stay loaded over a short wait, we might would expect the gun to become unloaded over a long sleep (if the would-be victim has any sense). This leads us to the conclusion that solving the persistence problem requires domain-dependent information about the actions involved. No such information exists in the standard representation of the shooting example, rendering it unsolvable.

3.6 Domain-Dependent Persistence

I believe that the compelling intuitive interpretation of the shooting example (where the victim dies) is a product of domain-dependent details furnished by the reader. One source of information uses communication maxims for puzzles, such as "the puzzle maker will include all information I need that I don't already have". This is a subtle source of confusion in the intuitive reading of many AI problems (Tweety flying, Missionaries and Cannibals, etc.) Since the shooting example is intended to be an example of general autonomous reasoning, we will avoid such inferences and concentrate on knowledge about the domain, as in the following simple argument for the persistence of loaded:

1. The reasoner is holding a loaded gun, and then waits.
2. No agent can unload the gun while it is held by the reasoner (assuming the reasoner will not unload it).
3. Therefore, the gun will not be unloaded during the wait.

Item 1) is the initial specification of the scenario, represented as before except that we have added that the reasoner is holding the gun:

$$\mathtt{holds}(\mathtt{hasgun}, 0) \wedge \mathtt{holds}(\mathtt{loaded}, 0) \wedge \mathtt{occurs}(\mathtt{wait}, 0)$$

Item 2) is a new domain-dependent rule that infers the non-occurrence of unload:

$$\forall m[\mathtt{holds}(\mathtt{hasgun}, m) \rightarrow \neg\mathtt{occurs}(\mathtt{unload}, m)]$$

From these axioms it is trivial to deduce item 3), $\neg\mathtt{occurs}(\mathtt{unload}, 0)$.

This simple example demonstrates how sufficient conditions can be used to prove action non-occurrences as well as action occurrences. Note that I did not use the traditional frame axioms, which would specify the persistence of all unrelated facts for every action, and therefore might suffer from a space blowup due to the number of such axioms. Instead, this approach exploits domain knowledge about the relationships between property change and action occurrence, and action occurrence and property values.

When such domain-dependent information is not available, the reasoner will not be able to make a prediction about that property change. This is to say that these causal theories may be incomplete, but we do not consider this to be a deficiency of our approach. Indeed, we should be more suspicious of approaches that grant omniscience no matter what is known about the domain [Lifschitz, 1987, Shoham, 1988]. The long-standing emphasis on domain-independent rules in causal reasoning is overly optimistic, leading to unwarranted inferences (such as the generation-minimization problem) when such rules are included.

3.7 Domain-Dependent Axioms

In this section I construct a formal framework of action based on a discrete time line that solves the persistence problem in a domain-dependent way. This framework consists of several different axiom forms that serve different roles in the reasoning: one form infers property changes from action occurrences: another infers action occurrences from property changes: yet another infers action non-occurrences from unsatisfied necessary conditions. and the last form infers action occurrences from temporally prior sufficient conditions. Together they solve a number of interesting problems in causal reasoning.

I use the notation of Chapter 2. including a negation function ($\mathbf{not}(p)$) on properties. defined by $\mathtt{holds}(\mathtt{not}(p).m) \equiv \neg\mathtt{holds}(p.m)$. To facilitate the representation of change. I first make the following predicate definition:

$$\mathtt{changes}(p.m) \equiv_{\mathrm{def}} (\neg\mathtt{holds}(p.m) \wedge \mathtt{holds}(p.m')) \qquad (3.11)$$

Note that to represent that a property goes from holding to not holding. we assert a change on its negation. i.e. it follows that:

$$\mathbf{changes}(\mathtt{not}(p).m) \equiv (\mathbf{holds}(p,m) \wedge \neg\mathbf{holds}(p.m')) \qquad (3.12)$$

Ramification axioms specify the effects of actions through the use of implication. where ϕ is a property and $\epsilon_1, \ldots, \epsilon_\nu$ are events:

$$\bigvee_{i=1}^{\nu} \mathtt{occurs}(\epsilon_i, m) \rightarrow \mathbf{changes}(\phi, m) \qquad (3.13)$$

Note that an instance of this form says nothing if ν equals 0, thereby making the disjunction false (by definition). Here, and in the rest of the forms. greek letters are used for terms that are replaced by domain-specific constants in instances of the forms. The above form could be rewritten as a list of forms in the following way:

$$\begin{gathered}\mathtt{occurs}(\epsilon_1.m) \rightarrow \mathbf{changes}(\phi.m)\\ \mathtt{occurs}(\epsilon_2.m) \rightarrow \mathbf{changes}(\phi,m)\\ \vdots\\ \mathtt{occurs}(\epsilon_\nu.m) \rightarrow \mathbf{changes}(\phi.m)\end{gathered}$$

This presentation is more verbose, but perhaps more readable. The choice is purely a matter of personal taste.

I also use axioms that specify when a particular property change must be due to an appropriate action. called *cause closure* axioms. These are similar to ramification axioms. except that the implication goes in the opposite direction, as expressed by the following form:

$$\mathbf{changes}(\phi, m) \rightarrow \bigvee_{i=1}^{\nu} \mathtt{occurs}(\epsilon_i, m) \qquad (3.14)$$

Note that ν could equal 0, in which case by definition the disjunction is false and the property can never be changed. Closure axioms were described by Len Schubert,[4]

[4] In a talk entitled "Solving the original frame problem without frame axioms or nonmonotonicity" at the 1988 meeting of the Society for Exact Philosophy.

and also constructed by Morgenstern and Stein for their causal reasoning framework [Morgenstern and Stein, 1988]. Interestingly enough, these axioms also look very similar to theorems that are added in Lifschitz's system when the **causes** predicate is circumscribed, e.g.

$$(\neg\mathtt{holds}(\mathtt{p}, s) \wedge \mathtt{holds}(\mathtt{p}, \mathtt{do}(a, s))) \rightarrow (a = \mathtt{a}_1 \vee a = \mathtt{a}_2 \vee a = \mathtt{a}_3)$$

Cause closure axioms perform the same role as traditional frame axioms: they are used to prove that particular properties are not changed by the occurrence of a particular action. However, this new form requires far fewer axioms (at worst the number of properties) and if actions are fairly independent then the disjunctions will be fairly short.

Together, ramification and cause closure axioms specify necessary and sufficient conditions for property change solely in terms of event occurrences. For conciseness, we can generally combine both implication directions into an equivalence, and I will call these combinations *change conditions*. For example, I might represent that my office door can only be opened by key or force in the following way:

$$\mathbf{changes}(\mathbf{open}, m) \equiv (\mathbf{occurs}(\mathbf{unlock}, m) \vee \mathbf{occurs}(\mathbf{bash}, m))] \qquad (3.15)$$

The above forms are sufficient to solve the persistence problem when the reasoning agent knows exactly which actions occur at each time, as in the idealized domains for the situation calculus where all property changes are intentionally performed by a single agent. In more realistic domains, there must exist mechanisms for inferring the occurrence or non-occurrence of events.

One way to deduce non-occurrence is through the absence of effects. For example, from $\neg\mathtt{holds}(\mathtt{open}, \mathtt{m}')$ and (3.15), we can infer $\neg\mathbf{occurs}(\mathbf{bash}, \mathtt{m})$. This is a valid and useful kind of inference to make, but it does not help solve the persistence problem since it is not a prediction. To solve the persistence problem, there must be axioms that support the derivation of $\neg\mathbf{occurs}(\mathtt{a}, \mathtt{m})$ based entirely on information no later than $\mathtt{m}$. This is the motivation behind the next axiom form, called *executability* axioms:

$$\mathbf{occurs}(\epsilon, m) \rightarrow \bigwedge_{i=1}^{\nu} \mathtt{holds}(\phi_i, m) \qquad (3.16)$$

This constitutes specifying necessary conditions for an event's occurrence, and when at least one such condition does not hold then the event cannot occur.

To infer when events *did* occur, the reasoner must have jointly sufficient conditions available. Interestingly enough, many previous causal theories do not provide such sufficient conditions, and therefore cannot infer occurrences unless explicitly told. This is usually due to the functional model of action [Lifschitz, 1987, Ginsberg, 1987, Haugh, 1987] where actions are "willed" by the reasoning agent, but also even in theories with a more general temporal model [Morgenstern and Stein. 1988].[5] To allow external events, it is essential that the sufficient conditions be expressed entirely in terms of properties, at least at some level. These *event enabling* axioms take the following form:

$$\bigwedge_{i=1}^{\nu} \texttt{holds}(\phi_i, m) \rightarrow \texttt{occurs}(\epsilon, m) \tag{3.17}$$

The properties in (3.16) and (3.17) are often termed *preconditions*, although usually it is unclear whether they are necessary, jointly sufficient, or both for the event's occurrence. Here I have made that distinction clear. Also, note that if the necessary and sufficient conditions are identical. they can be combined into an equivalence as we did with ramification and cause closure axioms.

When we have necessary and sufficient conditions for events as above, events are no longer "primitive" in the sense that an event's occurrence can always be rewritten in terms of properties that hold. Events, then, really just stand for convenient groupings of properties, allowing more concise and intuitive axioms. For example, instead of using the forms in (3.16) and (3.17), one could use a generation rule of the following form:

$$(\texttt{occurs}(\epsilon_1, m) \wedge \bigwedge_{i=1}^{\nu} \texttt{holds}(\phi_i, m)) \rightarrow \texttt{occurs}(\epsilon_2, m) \tag{3.18}$$

This is a shortcut for specifying sufficient conditions for ϵ_2 by augmenting those of ϵ_1. If the implication were in the reverse direction, the axiom would do the same with necessary conditions. In addition, event abstraction via implication [Tenenberg, 1988. Kautz and Allen. 1986] (i.e. $\forall m[\texttt{occurs}(\epsilon_2, m) \rightarrow \texttt{occurs}(\epsilon_1, m)]$) is a shortcut for specifying the inheritance of sufficient conditions. These relationships follow directly from this basic axiomatization of events in terms of properties. Note that this approach does not have the generation/minimization problem, simply because it does not use minimization of occurrences to infer that events do not occur.

Now I present the shooting problem using the above forms. The change conditions are:

$$\begin{aligned} \texttt{changes}(\texttt{not}(\texttt{loaded}), m) &\equiv \texttt{occurs}(\texttt{fire}, m) \vee \texttt{occurs}(\texttt{unload}, m) && (3.19)\\ \texttt{changes}(\texttt{alive}, m) &\equiv \texttt{occurs}(\texttt{fire}, m) && (3.20) \end{aligned}$$

For demonstration purposes, I have added the commonly included fact that the gun becomes not loaded through being fired. I specify necessary and sufficient conditions for `fire` in terms of a `pulltrigger` event, via the following rule (note the use of equivalence):

$$\texttt{occurs}(\texttt{pulltrigger}, m) \wedge \texttt{holds}(\texttt{loaded}, m) \equiv \texttt{occurs}(\texttt{fire}, \texttt{m}) \tag{3.21}$$

[5] Morgenstern and Stein do allow generation rules like (3.10), but one of the sufficient conditions must be another event, leading to a "chicken and egg" problem.

The conditions for `pulltrigger` need not be specified for this example. Given these domain axioms, we are interested on what can be predicted in the following scenario:

$$\mathtt{holds}(\mathtt{loaded}, 0) \land \mathtt{holds}(\mathtt{alive}, 0)$$

$$\neg\mathtt{occurs}(\mathtt{pulltrigger}, 0) \land \mathtt{occurs}(\mathtt{pulltrigger}, 1) \qquad (3.22)$$

Since `pulltrigger` did not occur at moment 0 (the more temporally general interpretation of the `wait` in the original version of the shooting problem), we can infer using (3.21) and (3.20) that holds(alive, 1). However, since we cannot ascertain the status of occurs(unload, 0), we cannot decide about holds(loaded, 1). Consequently, we cannot decide about occurs(fire, 1), and finally (as we have argued is proper) cannot predict whether holds(alive, 2) is true or not. Thus we have avoided wishful thinking.

For a more complicated example, we will consider adding appropriate domain dependent information so that we *can* prove that the victim is no longer alive a time 2. Recall the description in Section 3.5 of suspected intuitions behind the popular interpretation of the shooting problem: the gun can only be unloaded by the bearer, and because the bearer (the reasoning agent) has no reason to do so the gun will not be unloaded. There are several parts to the representation of the knowledge for this inference. I start with the change conditions (3.19) and (3.20) and the generation-like rule (3.21) from the previous example. I add another generation-like rule describing the `Unload` event, as follows:

$$\mathtt{occurs}(\mathtt{unload}, m) \equiv \exists a[\mathtt{holds}(\mathbf{hasgun}(a), m) \land \mathtt{holds}(\mathtt{fearsgun}(a), m)] \qquad (3.23)$$

We need another type of axiom, called a *domain constraint* that ensures that only one agent may have the gun at any given moment:

$$\mathtt{holds}(\mathbf{hasgun}(a), m) \land \mathtt{holds}(\mathbf{hasgun}(b), m) \rightarrow a = b \qquad (3.24)$$

We must also have some information about the **fearsgun** property function for both the reasoning agent (`me`) and the victim (`it`):

$$\neg\mathtt{holds}(\mathbf{fearsgun}(\mathtt{me}, m)) \land \mathtt{holds}(\mathbf{fearsgun}(\mathtt{it}, m)) \qquad (3.25)$$

The reasoning scenario is similar to before, except for the addition of assumed information about **hasgun**:

$$\mathtt{holds}(\mathtt{loaded}, 0) \land \mathtt{holds}(\mathtt{alive}, 0)$$

$$\mathtt{holds}(\mathbf{hasgun}(\mathtt{me}, 0) \land \mathtt{holds}(\mathbf{hasgun}(\mathtt{me}, 1)$$

$$\neg\mathtt{occurs}(\mathtt{pulltrigger}, 0) \land \mathtt{occurs}(\mathtt{pulltrigger}, 1) \qquad (3.26)$$

Step	Inference	Reason
a.	$\neg\texttt{holds(FearsGun(Me),}0)$	3.25
b.	$\texttt{holds(HasGun(Me),}0)$	3.26
c.	$a \neq \texttt{Me} \rightarrow \neg\texttt{holds(HasGun(}a\texttt{),}0)$	b,3.24
d.	$\neg\texttt{occurs(Unload,}0)$	a,b,c,3.23
e.	$\neg\texttt{occurs(PullTrigger,}0)$	3.26
f.	$\neg\texttt{occurs(Fire,}0)$	e,3.21
g.	$\neg\texttt{changes(}\overline{\texttt{Loaded}}\texttt{,}0)$	d,f,3.19
h.	$\texttt{holds(Loaded,}1)$	g,3.12
i.	$\texttt{occurs(PullTrigger,}1)$	3.26
j.	$\texttt{occurs(Fire,}1)$	h,i,3.21
k.	$\texttt{changes(}\overline{\texttt{Alive}}\texttt{,}1)$	j,3.20
l.	$\neg\texttt{holds(Alive,}2)$	k,3.12

Table 3.1: Proof showing causal derivation of $\neg\texttt{holds(Alive,}2)$

Given this scenario, we can infer $\neg\texttt{holds(Alive,}2)$, detailed by the abbreviated[6] proof in Table 3.1, where numbers refer to the above axioms and letters to proof steps.

Thus I have shown how to encode domain information so that our intuitions about the shooting problem are justified. The origins of this information seem to be a combination of knowledge about guns and conversational maxims about the problem statement being informative and not misleading. Details about the source of the knowledge, however, is beyond the scope of this thesis.

3.8 Nonmonotonicity and Domain-Dependence

Theories using the above framework are monotonic. If desired, we can add nonmonotonic rules in one of the many ways, e.g. using Reiter's default rules [Reiter, 1980]:

$$\frac{\texttt{holds(Loaded,}m) \wedge \texttt{holds(HasGun(Me),}m) : \mathbf{M}\ \texttt{holds(Loaded,}m')}{\texttt{holds(Loaded,}m')}$$

Literally, this rule says "if you are holding a loaded gun and it is consistent that it is still loaded at the next time point, then assume it will still be loaded". Another way to phrase this is "you would (probably) know if a gun became unloaded while you were holding it", in the sense that provability means knowledge. There are many uses for this type of default rules.

[6] Abbreviated in the sense that the implicit inference rules are more complex than but reducible to modus ponens.

Condoning the use of nonmonotonicity may appear to contradict my arguments against wishful thinking, but this is not actually the case. It is the *domain-independent* use of nonmonotonicity that I criticize. Nonmonotonic rules such as the above make claims about particular properties, i.e. that these properties will usually persist. How often the default rule must be right depends on how costly it is to be wrong, and the cost of ascertaining a more accurate evaluation of that property. There is no doubt, though, that there are cases where the use of such rules is justifiable.

3.9 Chapter Summary

The traditional approach to the frame problem has been to look for some domain-independent constraints on causal reasoning situations that infer the persistence of fluents by default. This approach is fraught with philosophical and technical problems that question its fundamental soundness. Instead, the reasoner can succinctly axiomatize the domain-dependent information needed to infer that a fluent persists, and reason about persistences in the same manner as changes. A short version of this chapter appears in Weber [1989a].

Chapter 4

Default Logic and the Qualification Problem

The previous chapter frequently used "causal rules" to infer action occurrences (and non-occurrences. to solve the frame problem). The antecedents of these rules consisted of a list of sufficient conditions (usually called *preconditions*, all of which must be proven true in order the infer the action occurrence. The *qualification problem* says that in many domains. this list will be so long as to make its evaluation impractical; yet agents are still obliged to make predictions about the occurrence.

McCarthy [1977] introduced the most famous example of the qualification problem. called the "potato in the tailpipe" scenario. Suppose you get in your car, and turn the key. You *might expect* the car to start, despite not knowing (for sure) whether the battery is still charged, whether the ignition system is intact, whether there is a potato in the tailpipe, nor whether any of a long list of relevant fluents hold. The qualification problem asks how a reasoner can make a causal prediction based on a *reasonable* body of evidence. This chapter examines some proposed solutions to this problem, pointing out strengths and weaknesses, and concludes with a generalization of the use of default logic in solving the qualification problem.

4.1 The Qualification Problem*s*

The qualification problem actually consists of several levels [Elgot-Drapkin *et al.*, 1987, Ginsberg and Smith, 1987]. On the one hand, the list of preconditions may be infinitely long, or at least so large that a reasoning agent with a limited lifetime would find it impossible to completely examine all elements. Even if it is possible to examine the entire list of preconditions, a reasoning agent may find it impractical to do so in reasoning situations due to time limitations. And even if the list is of manageable length, it may be impractical to determine the truth of all the preconditions. For example, it would be a chore to check the tank for gas, the engine for timing, and the tailpipe for potatoes whenever starting a car, even if those were the only preconditions.

Most work on the qualification problem has concentrated on the last version, i.e. assuming the list of preconditions is of manageable length, how can we avoid the explicit determination of the truth values of the preconditions? To answer this, these approaches employ some technique for non-monotonic reasoning that assigns default truth values to the majority of the preconditions, called *qualifications*. Due to the nature of defaults, contradictory information about these qualifications will override the defaults, and change the prediction.

4.2 Minimizing Disability

McCarthy [1977] proposed a solution to this dilemma, which is similar to a later solution due to Shoham [1988]. The solution is to assume that an action is successful if its important preconditions hold, using rules such as:

$$\mathtt{ignition}(s) \wedge \neg\mathtt{disabled(start},s) \rightarrow \mathtt{running(do(start},s) \qquad (4.1)$$

In words, if the ignition is engaged and the car-starting action is not disabled, then the car will be running after the action is applied. The $\mathtt{disabled}$ is circumscribed [McCarthy, 1980], a.k.a *minimized*, to implement the assumption that actions are not normally disabled. Thus, when lacking any evidence to the contrary, the reasoner will assume $\neg\mathtt{disabled(start,s)}$ for any situation $\mathtt{s}$, making knowledge of $\mathtt{ignition(s)}$ imply the conclusion $\mathtt{running(do(start,s)}$. Exceptions to the successful starting of the car are captured by individual axioms implying disability, e.g.

$$\begin{array}{c} \neg\mathtt{charged}(s) \rightarrow \mathtt{disabled(start},s) \\ \mathtt{potato}(s) \rightarrow \mathtt{disabled(start},s) \end{array} \qquad (4.2)$$

For example, suppose $\mathtt{ignition(s)}$ is known when attempting to start a car while in situation $\mathtt{s}$. If neither $\mathtt{potato(s)}$ nor $\neg\mathtt{charged(s)}$ are known to hold in $\mathtt{s}$, then circumscription will produce the conclusion $\neg\mathtt{disabled(start,s)}$. When this fact is taken with (4.1), and the knowledge about the ignition, the reasoner concludes $\mathtt{running(do(start,s)}$. In addition, by the contrapositives of the rules in (4.2), the reasoner also may conclude $\neg\mathtt{potato(s)}$ and $\mathtt{charged(s)}$. However, if the reasoner comes to know $\mathtt{potato(s)}$, then by the latter rule in (4.2) it may conclude $\mathtt{disabled(start,s)}$, and therefore may no longer conclude $\mathtt{running(do(start,s)}$ (nor may it conclude $\neg\mathtt{running(do(start,s))}$). Incidentally, the reasoner is no longer allowed to conclude $\mathtt{charged(s)}$. The fact that knowledge about the tailpipe overturned a conclusion about the battery is quite odd. This is due to the fact that qualifications are given defaults as a side-effect of assumptions about actions, instead of through assumptions about the typical value of the qualifications themselves. This "minimizing disability" approach has a number of additional problems, detailed in the following sections.

4.2.1 Not the shooting problem, again

When the minimizing disability technique is applied to a series of actions. multiple minimal models are produced in similar manner to Hanks and McDermott's shooting problem (cf. Section 3.2). For example. suppose you are working on the car and decide to disconnect the battery from the ignition system. Removing a lead will probably accomplish this. although the qualification problem says we cannot be sure. This can be represented by the rule:

$$\neg\mathtt{disabled}(\mathtt{disc}, s) \rightarrow \neg\mathtt{charged}(\mathtt{do}(\mathtt{disc}, s))$$

where "disc" stands for the action of disconnecting the lead. So given the knowledge $\mathtt{charged(s)}$. the reasoner defeasibly concludes $\neg\mathtt{charged(do(disc, s))}$ when the disabled predicate is minimized. Now suppose we try to start the car, knowing $\mathtt{ignition(do(disc. s))}$ and the axioms in (4.1) and (4.2). Intuitively, we would expect the car starting action to be disabled, but this is not the case in all disabled-minimal models. since in some models $\neg\mathtt{disabled(disc, s)}$ is true and in others $\neg\mathtt{disabled(start. do(disc, s))}$ is true. After disability is minimized the reasoner cannot conclude either of $\neg\mathtt{charged(do(disc, s))}$ or $\mathtt{running(do(start, (disc, s)))}$, since there are no models where both are true.

4.2.2 Defaults out of context

The minimizing disability approach also has a subtle problem stemming from the fact that a qualification of a particular action defaults to false even when that action is not being performed. For example, suppose you have one of those cars that complains when the ignition is engaged and the seat belts have not been fastened. If your routine is such that you rarely have the seat belt on when starting the car, then "seat belt on" is a qualification of the automatic complaint buzzer. Using the disability representation, this could be written as:

$$\mathtt{ignition}(s) \wedge \neg\mathtt{disabled}(\mathtt{complain}, s) \rightarrow \mathtt{buzz}(\mathtt{do}(\mathtt{complain}, s))$$

$$\mathtt{seat\text{-}belt\text{-}on}(s) \rightarrow \mathtt{disabled}(\mathtt{complain}, s) \qquad (4.3)$$

However, when the disabled predicate is minimized, $\mathtt{seat\text{-}belt\text{-}on}(s)$ defaults to false for all situations. This was not the intent of the rule, which was supposed to assert that the seat belt is rarely on *when starting the car*, not over arbitrary situations. Thus qualifications should really be evaluated with respect to the context established by the regular preconditions for the action. In Section 5.3.3 I show a simple way of evaluating qualifications in context using conditional statistics.

4.2.3 Exceptions to exceptions

Ginsberg and Smith [1987]. noted that a problem with this approach is that exceptions to action occurrence, such as potatoes in tailpipes, may themselves have exceptions, such as a special exhaust system with two tailpipes. In other words. the sufficient conditions for concluding disability also have the qualification problem. A solution to this problem would be to allow disability terms in the sufficient conditions for disability. as in:

$$\mathtt{potato}(s) \wedge \neg\mathtt{disabled(potato\text{-}effect,}s) \rightarrow \mathtt{disabled(start,}s)$$

$$\mathtt{dual\text{-}tailpipe}(s) \rightarrow \mathtt{disabled(potato\text{-}effect,}s)$$

Informally. it is usually true that the unusual appearance of a potato in a tailpipe actually keeps the car from running. Of course. the exception to the exception may *itself have an exception*, to be handled in the same fashion.

Now consider how the disabled terms interact in the above example. If potato(s) is not known, then both **disabled(start,s)** and **disabled(potato-effect,s)** default to false during circumscription. However, if potato(s) later becomes known to be true, then we must choose whether the potato actually disables car-starting, or some other factor renders the potato ineffective. This choice can be written as the following disjunction:

$$\neg\mathtt{disabled(potato\text{-}effect,s1)} \vee \mathtt{disabled(start,s1)}$$

This disjuction becomes a theorem, resulting in conflicting defaults a.k.a multiple minimal models. This means that the disabled predicate is not completely determined by circumscription; in particular, disabled(start,s) is not a theorem nor a non-theorem. meaning that the reasoner cannot predict the truth of **running(do(start,s))**. This runs contrary to the intuition behind the axioms, which says that if there is a potato in the tailpipe, then assume it will stop the car from running.

4.3 Qualifications and Ramifications

Ginsberg and Smith [1987] observed an interesting relationship between the qualification, frame, and ramification problems. If an effect of an action contradicts the persistence of a known domain fact, either 1) the action was in fact qualified by the domain fact and the effect is not achieved, or 2) the negation of the domain fact is an additional effect of the action. For example, if a car cannot be running and yet have a potato in its tailpipe, and there is a potato in the tailpipe of a car that is attempting to start, then either 1) the car will not start, or 2) the car will start and in so doing remove the potato. Which case will occur depends on additional domain knowledge, e.g. an axiom that says that the potato will be expelled if and only if it does not fit snugly.

Any qualification can therefore be represented as a conflict between a persistence default and a qualified action effect with respect to some known constraint about the domain. Ginsberg and Smith described a technique whereby each action is associated with a set of constraints upon which its effects may differ. The reasoner compares the expected outcomes with and without these constraints, and a prediction is made based on what they have in common. They call these expected outcomes *closest possible worlds*, after the semantic notion from the theory of counterfactuals [Ginsberg, 1986]. However, as Winslett [1988] points out, they try to capture the semantic notions of "closest" worlds and what worlds "have in common" through crude syntactic comparisons such as the number of identical formulas in axiomatizations. For these reasons, their techniques come across as either vague or *ad hoc*, not really providing a versatile, testable solution to the qualification problem.

4.4 Circumscribing Qualifications

In his influential paper "Formal Theories of Action" Lifschitz [1987] described a solution to what he called the qualification problem. Actually, he solved a more limited problem quite different from what was addressed by the preceding schemes. I describe it here for the sake of clear comparisons with the terminology in the literature.

The above solutions divided up domain fluents known to be preconditions between those that have reasonable default values and those that must be checked explicitly. Lifschitz presented a representation that divided up fluents into those known to be preconditions and those assumed to not be related as preconditions. Recall that his Law of Change stipulated that for an action to imply an effect, all preconditions of that action must be true. A fluent f is asserted as a precondition of action a using the literal $\mathbf{precond}(f, a)$. In order to prove that *all* preconditions of an action are true, however, we require assertions about what fluents are *not* preconditions, i.e. negated literals like $\neg\mathbf{precond}(f, a)$. Lifschitz's approach provides the necessary negated **precond** literals by circumscribing the **precond** predicate. This is better than completely specifying the **precond** predicate explicitly as a list of positive and negative literals, although compact representations can be constructed that avoid the need for minimization (e.g. adding $\mathbf{precond}(p, \mathbf{a}) \equiv (p = \mathbf{q} \vee p = \mathbf{r})$).

Thus the two kinds of solutions to the "qualification problem", and therefore the problems they solve, are orthogonal. The solutions could be combined, where Lifschitz's solution defines the preconditions and McCarthy's solution imposes the finer distinction of which preconditions are given defaults; the notations, however, are different enough that it would take some careful formalization to combine them yet retain the perceived advantages of each. Besides, in Chapter 5 I show how the use of statistical inference subsumes both solutions, in the following way: statistical independence makes Lifschitz's distinction between preconditions and non-preconditions, and the values of conditional statistics suggest degrees of belief in qualifications instead of defaults.

4.5 Defaults and Qualifications in General

It is more appropriate to view logical solutions to the qualification problem as representational rather than computational, especially when semantic notions such as possible worlds or model elimination are used to capture defaults. The above solutions often try to capture performance issues, e.g. by writing the axioms so that a mixture of forward and backward chaining can be used, but it is not clear if abstract measures such as numbers of facts actually dominate the computational fluents at the unspecified algorithmic level. Proof theory does not easily support computational claims without the underlying mechanism for inference; this is especially true for nonmonotonic entailment.

This section dispenses with claims that axioms should be written this way or that way and examines how default logic can be used to solve representational and inferential aspects of the qualification problem. Computational aspects will ignored until Chapter 6, which discusses algorithmic heuristics for statistical causal reasoning.

4.5.1 Default logic and qualifications

Consider adding a defeasible version of the implication operator to first-order logic, in order to use the sentence $\phi \rightsquigarrow \psi$ to mean "the sentence ϕ defeasibly implies ψ". The formal properties of such a task have been investigated in numerous places [Loui, 1987b, Nute, 1986, Reiter, 1980]; here we will rely on the following informal definition: if $\phi \rightsquigarrow \psi$ is a default rule, ϕ is considered true by the reasoner, and ψ is consistent with *everything* considered true, then ψ is also considered true. This is the same reading as the more familiar but notationally awkward default rule of Reiter [1980]:

$$\frac{\phi : \mathbf{M}\,\psi}{\psi}$$

This operator is used to solve the qualification problem by writing causal rules which capture reasonable conclusions from appropriate evidence, e.g. for car starting, using an explicit time line:

$$\mathtt{ignition}(m) \rightsquigarrow \mathtt{running}(m') \tag{4.4}$$

$$\mathtt{ignition}(m) \land \neg\mathtt{charged}(m) \rightsquigarrow \neg\mathtt{running}(m') \tag{4.5}$$

In words, if the car is not running and the key is turned, then it is reasonable to assume the car will be running at the next moment; however, if the battery is dead, then it is reasonable to assume that the car will *not* be running at the next moment. Since the conclusions of these two rules conflict, the prediction about the car starting depends on the order that the rules are applied (in Reiter's terminology, there will be *multiple extensions* of the rule set). If both **ignition**(1) and $\neg$**charged**(1) have been asserted, then when (4.4) is invoked, **running**(2) will be concluded. This new fact blocks the invocation of (4.5), since $\neg$**running**(2) is no longer consistent: the reverse happens if (4.5) is applied first. Reiter suggested that the conclusions of a set of default rules are those sentences true in *all* extensions. This stipulation is not appropriate for our purposes, since no qualified predictions would be true in all extensions, and therefore no predictions would be made. Since in general there will always be exceptions to causal rules, and therefore conflicting default rules, we must derive a conclusion even in the face of this ambiguity.

4.5.2 Resolving conflicting defaults through specificity

Intuition tells us that that the application of default rule (4.5) should be preferred to the application of (4.4), since (4.5) involves more knowledge about the situation: if we know a fact such as potato(1), then we should prefer rules that use this knowledge over ones that don't. This intuition is captured by the principle of *specificity* [Bacchus, 1988] a.k.a. *inferential distance* [Touretzky, 1984, Etherington, 1987]. To use another more familiar example, suppose that Tweety is a penguin and therefore also a bird; we would prefer to use the rule that most penguins don't fly over the rule that most birds do fly. In general, we can define the following principle:

Heuristic 4.1 (Specificity for Default Logic) *Given two conflicting default rules $E \leadsto h$ and $F \leadsto \neg h$, if E entails F with respect to some background knowledge (and the reverse is not true), E is said to be more specific and the rule $E \leadsto h$ should be applied first.*

I should note that there are many cases where conflicting defaults are not resolved by appealing to specificity. A famous example is the "Nixon diamond", where the reasoner has the following defaults rules:

$$\mathbf{Quaker}(x) \leadsto \mathbf{pacifist}(x)$$

$$\mathbf{Republican}(x) \leadsto \neg\mathbf{pacifist}(x)$$

The default truth of **pacifist(Nixon)** depends on the order of application of these rules, since we know both **Quaker(Nixon)** and **Republican(Nixon)**. We cannot apply specificity to resolve this conflict, since their antecedents are not logically related; there must be some other mechanism that disambiguates these defaults. However, if there is also the default rule:

$$\mathbf{Republican}(x) \wedge \mathbf{Quaker}(x) \leadsto \neg\mathbf{pacifist}(x)$$

then the conflict is resolved, since by specificity this rule is applied before both other rules, and the first becomes blocked and the second redundant.

4.5.3 How default logic solves the qualification problem

This general view of default rules for causal reasoning makes a solution to the qualification problem much more apparent. Recall the following two default rules about car-starting:

$$\mathbf{ignition}(m) \rightsquigarrow \mathbf{running}(m')$$

$$\mathbf{ignition}(m) \wedge \neg\mathbf{charged}(m) \rightsquigarrow \neg\mathbf{running}(m')$$

The first rule *implicitly* captures that the car's battery is seldom dead. Otherwise, if the battery often died, then it would not be a reasonable assumption that the car will start when the ignition is engaged.

In other words, a default rule makes implicit default assertions about the sufficient preconditions that do *not* appear in the antecedent. When these other preconditions become known, however, the reasoner is forced to use a more specific default rule. This is the essence of all the default logic solutions to the qualification problem. An analogue of this feature will also be crucial to my statistical solution to the qualification problem, discussed in Section 5.3.

4.5.4 The lottery paradox

A classic problem of default logic approaches is the *lottery paradox* [Kyburg, 1970]. Imagine reasoning about a lottery where one of a large number of people will win a prize. The chance that any particular person will win is certainly very small, making it a reasonable default that the person will not win. Since this can be uniformly applied to all people involved, the conjunction of all the defaults implies that no person will win. This contradicts the definition of the lottery, which says that there will be a winner. The reason for this contradiction is that defaults do not lose force when composed, yet the statistical facts they reflect do.

The lottery paradox appears in the use of defaults for causal reasoning. The above car-starting default rules allow the reasoner to assume, for a particular car-starting instance, that the car will start and therefore the battery will not be dead. This seems perfectly reasonable. However, when taken together, these particular assumptions imply that since the reasoner has no future counter-evidence, the car will *always* start and the battery will *never* be dead. This would indicate that battery cables or preventive replacement of batteries are superfluous, and other conclusions we know to be wrong in practice. To correct this problem, it has been suggested that the facts in the antecedent of a default rule must not be true by default themselves; this would prevent far-reaching predictions, but also prevent reasonable predictions even two moments ahead. Shoham [1988] suggested attaching a number to each default rule (actually the projection of a "potential history") that bounds the number of times the rule can be applied in a chain. This solution is too *ad hoc* for examples such as the car battery, where there is no most reasonable cutoff period (although things do tend to break as soon as the warranty expires).

4.6 Chapter Summary

A causal reasoner will not have complete knowledge about a realistic domain. Solving the qualification problem involves making causal predictions in the face of this ignorance. Default rules provide a mechanism for making reasonable predictions from limited information, thereby addressing the qualification problem. However, since default rules are forced to round "usually" to "always" valuable information is lost and inconsistencies may develop in the resulting theory. A condensed version of this chapter (and the next chapter) appears in Tenenberg and Weber [1989].

Chapter 5

Statistical Causal Inference

A default rule works well when its antecedent implies that its consequent is almost always or almost never true, because it assigns one of two truth values, true or false. Default rules do not handle the middle ground, where the antecedent provides information about the consequent, but not to the point of declaring it true or false. For example, suppose the car is known to have difficulty starting in cold weather: certainly `cold(0)` is an important fact to consider when predicting `running(1)`, but to say that the car will start anyway or definitely not start is inappropriate.

This chapter describes a generalization of the default logic approach that derives a *degree* of belief in the consequent (effect) given the knowledge that the antecedent (precondition list) is true. This degree can be based on actual domain observations about the relative frequency of the effect given the preconditions, e.g. the car starts 70% of the time in cold weather. This allows for much more expressive relationships between preconditions and effects. It also captures the "non-monotonic flavor" of the qualification problem. For example, the car may start 95% of the time that the key is turned, only 70% of the time the key is turned and the weather is cold, but back up to 90% of the time the key is turned, the weather is cold, and the anti-freeze is extra strength.

There may be multiple relative frequencies applicable to the same prediction. In the percentages above, if the reasoner ignores the fact that the weather is cold, it may derive a 95% degree of belief instead of the 70% degree of belief. This is the *reference class problem* [Kyburg, 1983] that has concerned philosophers and statisticians for several decades. A reference class is simply a set containing the object in question; this set is used to suggest a degree of belief that the object belongs to another set. Thus a reference class can be thought of as "relevant evidence", analogous to the antecedent of a default rule. There may be multiple reference classes that suggest different degrees of belief; a commonly accepted principle is to use the more *specific* reference class. This is the analogue of the rule preference from default logic described in Section 4; in fact, specificity was discussed for degrees of belief before default logics were even invented.

This approach does not suffer from the lottery paradox as does the default logic approach, since degrees of belief do not need to be "subsidized" in order to be as strong as true or false. For example, given the belief that the car starts 95% of the time, the reasoner derives that there is a 95% · 95% ≈ 90% chance that the car will start twice[1], a 85% chance for three times, etc. This captures the intuition that the more times an action is attempted, the more likely it will fail during one of those attempts. This commonsense fact is totally lost in the default logic approach.

As with default logic, the essence of how this approach solves the qualification problem comes from what a rule says about those preconditions that do *not* actually appear in the rule. For example, if it is believed that the car starts 95% of the time, and it is believed that a dead battery prevents the car from starting (all of the time), then it must also be believed that the battery is dead less than 100% − 95% = 5% of the time. In fact, all the situations that would prevent the car from starting are believed to total exactly 5%. Thus these causal rules capture *statistical generalizations* about the combined effect of an action's qualifications.

5.1 Basic Statistical Notation and Inference

This section outlines the statistical notation to be used throughout the rest of this thesis. The notation is modeled after that of Kyburg [1974, 1987], and I refer the interested reader to his work for a more detailed and rigorous treatment of the technical and philosophical issues involved in the representation of uncertainty. The notation to follow will be sufficient to convey the concepts involved in this treatment of uncertainty and causality.

5.1.1 Statistics and basic theorems

Kyburg makes an important distinction (which is often ignored in AI work [Dean and Kanazawa, 1988]) between a *statistic*, which captures the relative frequency between sets of objects, and a *probability* that a *particular* object is a member of a set with respect to some knowledge about that object. These two concepts are often confused because they both assign a value to similar syntactic objects from the interval [0, 1] within a field of numbers. However, it is important to bear in mind the difference: a statistic is a generalization over a class of objects, whereas a probability is a belief about a particular object.

A statistic is written using the "`%`" predicate, e.g. "`%(fliers | birds)` = .95" means that 95% of the objects in the set of birds also belong to the set of fliers (I use the conditional bar (|) instead of comma (,) for this predicate to keep the order

[1] Assuming that the car starting trials are independent. If they are not, information about how they are dependent can be used to combine the individual chances.

of arguments clear). In the case of finite sets, a statistic can be defined in terms of cardinality (the predicate "#"), in the following way:

$$\%(a \mid b) = \frac{\#(a \cap b)}{\#(b)}$$

provided $\#(b) \neq 0$. This definition of statistics, together with standard axioms from set-theory, is sufficient to prove the following traditional "Laws of Probability" for statistics:

Range:	$0 \leq \%(a \mid b) \leq 1$
Additivity:	if $\%(a \cap b \mid c) = \emptyset$, then $\%(a \cup b \mid c) = \%(a \mid c) + \%(b \mid c)$
Chain rule:	$\%(a \cap c \mid b) = \%(a \mid b) \cdot \%(c \mid a \cap b)$

The chain rule can also be written as the familiar *Bayes Theorem*:

$$\%(c \mid a \cap b) = \frac{\%(a \cap c \mid b)}{\%(a \mid b)} = \frac{\%(c \mid b)}{\%(a \mid b)} \cdot \%(a \mid c \cap b)$$

which shows how to "swap" sets on different sides of the conditional bar.

5.1.2 Rational corpora, probabilities, and specificity

A set of these statistical assertions, plus other sentences in first-order logic (with axiomatic set theory, identity, and arithmetic operations) comprise an agent's *rational corpus*. This corpus can be used to derive a probability for a ground sentence that is equivalent to asking if an object is a member of a set, e.g. we might need a degree of belief for the sentence "$\texttt{Tweety} \in \texttt{fliers}$." Formally, if ϕ is a sentence and KB is a rational corpus, probability $\mathrm{P}(\phi \mid \mathrm{KB}) = p$ iff there exist terms x, y, z such that:

1. "$\varphi \equiv x \in z$" is a sentence in KB,

2. "$x \in y$" is a sentence in KB,

3. "$\%(z \mid y) = p$" is a sentence in KB, and

4. x is a random element of y w.r.t z relative to KB.

For example, suppose an agent's KB contains only the logical consequences of the following facts:

$$\%(\texttt{fliers} \mid \texttt{birds}) = .95$$

$$\texttt{Tweety} \in \texttt{birds}$$

then $P(\texttt{Tweety} \in \texttt{fliers} \mid KB) = .95$ since condition 1 is trivially satisfied[2], binding z to fliers, condition 2 binds y to birds, condition 3 defines the result, and condition 4 is true because *as far as the agent knows*, x could be any element of y. The set y is what we have been calling the "reference class". As remarked earlier, the choice of reference class can change the resulting degree of belief.

It is interesting to note that condition 4 above subtly imposes the principle of specificity, i.e. a statistic with a more specific reference class is preferred when assigning probabilities. To see how, consider adding the following sentences to the KB:

$$\%(\texttt{fliers} \mid \texttt{penguins}) = .005$$

$$\texttt{Tweety} \in \texttt{penguins}$$

$$\texttt{penguins} \subset \texttt{birds}$$

Now our previous derivation of .95 no longer holds since condition 4 is violated: Tweety is no longer an *epistemically random*[3] element of birds, since it is now a belief that Tweety is in penguins. Condition 4 is satisfied for the reference class penguins, so the resulting degree of belief is .005.

As with default logic (cf. Section 4.5.2), this notion of specificity does not choose between some competing statistics. An analogous version of the "Nixon Diamond" can be constructed for statistical beliefs, as follows:

$$\%(\texttt{pacifists} \mid \texttt{Quakers}) = 19$$

$$\%(\texttt{pacifists} \mid \texttt{Republicans}) = 1/9$$

$$\texttt{Nixon} \in (\texttt{Quakers} \cap \texttt{Republicans})$$

Specificity does not say which statistic to use to establish a belief in "$\texttt{Nixon} \in \texttt{pacifists}$", unless we also have a statistic that conditions pacifists on both Quakers and Republicans, in which case this more specific statistic is preferred.

These definitions comprise a statistical interpretation of probabilistic belief. There are other interpretations of probability, but a statistical approach is the choice of this thesis because it emphasizes that the meaning of the probabilities is grounded in distributional facts about the domain. It also suggests that the statistical beliefs of the agent might be derived from empirical observations about the relationships between domain properties.

[2] All examples in this thesis use ϕ's of the form "$x \in a$", so condition 1 will always be trivially satisfied.

[3] This term is due to Josh Tenenberg, who also contributed substantially to the development of the ideas in this chapter.

5.1.3 Considerations other than specificity

Actually, Kyburg's rational corpora contain more general statistical assertions of the form:

$$\%(z \mid y) \in [p, q]$$

which is taken to mean "the proportion of y's that are z's is in the *interval* from p to q", where $p \leq q$. Statistical assertions of this form can be used in the obvious way to capture a belief that the proportion of birds that fly is somewhere between .9 and .95. Two special cases of this form are important: when $p = q$ the assertion is equivalent to asserting a point value, and when $p = 0$ and $q = 1$ the assertion is a tautology and therefore conveys no information. Kyburg also presents a stronger version of specificity, where the relative vagueness of the intervals is considered in addition to the relative specificity of the reference classes. For example, if instead I told you that `Tweety` was a "speckled nuthatch" you might not have a statistic about flying that has a reference interval more informative than $[0, 1]$, and it would be better to just use the statistic conditioned just on birdhood. More formally, his modified principle states that statistics for a reference set Y will always be preferred to a reference set W whenever $Y \subset W$, as long as the reference interval $[y_1, y_2]$ for Y is not weaker than the reference interval $[z_1, z_2]$ for Z, i.e. $[z_1, z_2] \not\subset [y_1, y_2]$. Loui [1987b] has successfully implemented a system that applies this principle to retrieve the interval probability that reflects the statistic with the most appropriate reference class.

In the interest of simplicity, this thesis will largely ignore the advantages of intervals and the more complex version of specificity, as well as other limitations of specificity [Kyburg, 1989]. I will assume that the reasoning agent has enough statistical knowledge to reason effectively with the point-valued statistical assertions that follow from her experience in the domain.

5.1.4 Odds and belief

An *odds statistic* $\$(h \mid r)$ or just *odds* is defined in terms of relative frequency to be the following:

$$\$(h \mid r) = \frac{\%(h \mid r)}{\%(\overline{h} \mid r)} = \frac{\%(h \mid r)}{1 - \%(h \mid r)}$$

provided, of course, that $\%(\overline{h} \mid b) > 0$. Thus *odds values* are rationals in $[0, +\infty)$; a value from $[0, 1)$ says that more $\overline{\mathtt{h}}$'s are $\mathtt{r}$'s than are $\mathtt{h}$'s; a value in $(1, +\infty)$ says the opposite. This relationship between odds values and frequency values is depicted in Figure 5.1. I define the *degree of belief* predicate $\mathtt{B}$ to be:

$$\mathtt{B}(x \in h \mid x \in r) = \frac{\mathtt{P}(x \in h \mid x \in r)}{\mathtt{P}(x \in \overline{h} \mid x \in r)} = \frac{\mathtt{P}(x \in h \mid x \in r)}{1 - \mathtt{P}(x \in h \mid x \in r)}$$

That is, beliefs are to probabilities as odds are to relative frequencies.

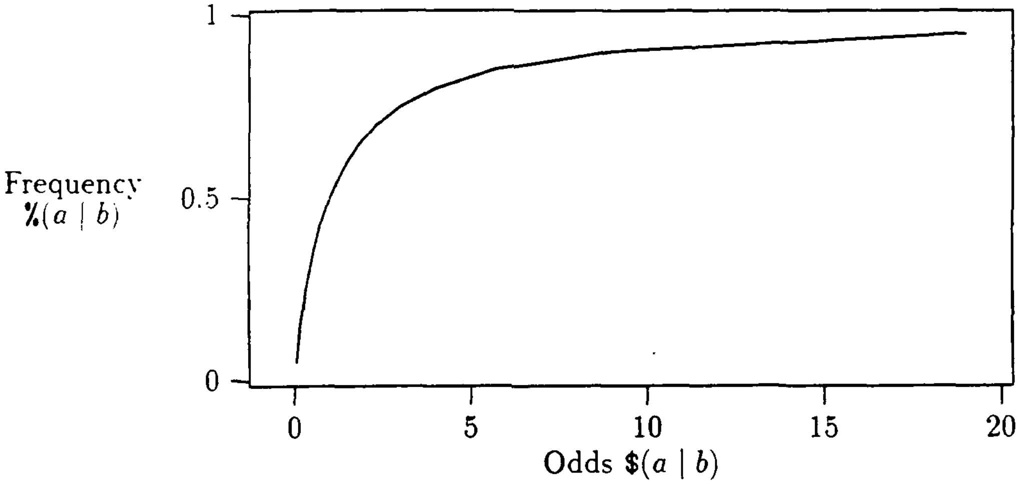


Figure 5.1: The relationship between the odds and relative frequency statistical values

Note that the odds statistic carries no less information than the relative frequency statistic, since

$$\%(h \mid r) = \frac{\$(h \mid r)}{1 + \$(h \mid r)}$$

and therefore it is not surprising that frequencies and odds have very similar properties. I will usually use the odds formulation, especially in examples, since the numbers involved tend to be more intuitive, being so similar to the common notion of odds making, e.g. "I'll be you ten to one that it isn't true". When it does not matter which formulation is used, I will use the term "statistic".

5.1.5 Conditional Independence

Definition 5.1 (Conditional Independence) *Two sets a and b are* conditionally independent *given a third set c if and only if* $\%(a \mid b \cap c) = \%(a \mid c)$.

This statistical constraint is also equivalent to each of the following:

$$\mathtt{I}(a, b, c)$$

$$\%(b \mid a \cap c) = \%(b \mid c)$$

$$\%(\neg a \mid b \cap c) = \%(\neg a \mid c)$$

$$\$(a \mid b \cap c) = \$(a \mid c)$$

In words, if you are already considering that an object is a c, then knowing that it is a b does not bear on it being an a, and vice versa. Systematic information about independence allows us to only represent a subset of the statistics needed for reasoning, since the remaining statistics will be equal to or derivable from the subset. For example, independence can be used to derive the frequency for a conjunction, since together with the chain rule:

$$\%(a \cap b \mid c) = \%(a \mid c) \cdot \%(b \mid a \cap c) = \%(a \mid c) \cdot \%(b \mid c)$$

This fact is often presented as a definition of independence; while being equivalent, it is more numerical than the definition given above, and therefore less intuitive.

5.1.6 Acceptance and the lottery paradox

It is sometimes useful to speak of the set of sentences that an agent believes with enough strength to be considered certain for all practical purposes. The process that defines the set of *practical certainties* is called *acceptance*. A simple mechanism for acceptance involves applying a threshold to beliefs, although a better acceptance mechanism would apply different thresholds depending on the context: for example, one should be very certain that a gun is not loaded when cleaning it.

Inference is performed on the statistics, which derive beliefs, and then accepted sentences become practical certainties. It is important to note that inference is not generally performed within the set of practical certainties: this fact provides an answer to the lottery paradox (cf. Section 4.5.4), since it is entirely consistent that "x will not win" is a practical certainty for all participants x, yet "someone will win" is also a practical certainty.

5.2 Statistics as Causal Rules

Having now outlined the basics of statistics and probabilities, I turn to their application to causal reasoning. In Chapter 2, I described a causal notation that represented fluents as sets of moments, and then used standard set operators to combine fluents. For example, ignition is the set of all moments at which the car's key is turned. I represent that a fluent is true of a particular moment through set membership, e.g. $\texttt{t} \in \texttt{ignition}$. Since I have described a statistical notation using sets, it will be convenient to use the set-based causal notation when combining the concepts of statistical inference and causal reasoning.

An agent's ***domain knowledge*** consists of set membership assertions such as $1 \in \texttt{ignition}$. This knowledge represents what is known about the domain at a particular time of interest. An agent's ***statistical causal knowledge*** consists of conditional statistical assertions about fluents. This knowledge represents what is known about the domain in general. Specifically, I will use statistical assertions about how fluents that contain a moment m influence whether other fluents contain m', e.g.

$$\%(\{m : m' \in \texttt{running}\} \mid \texttt{ignition}) = p \tag{5.1}$$

That is, the proportion of time points in which the car is running preceded by a time point in which the key was turned is equal to p. Since I will be using rules of this form extensively, I will use the shorthand notation X^{*} for the set $\{m : m' \in X\}$.

5.2.1 Causal probabilities

The union of the domain knowledge with the statistical causal theory yields a rational corpus from which we may derive probabilities that moments belong to fluents. For example, suppose the KB contains only the logical consequences of the following sentences:

$$\%(\mathtt{running}^{*} \mid \mathtt{ignition} \cap \overline{\mathtt{running}}) = .95 \tag{5.2}$$

$$0 \in \mathtt{ignition}$$

According to principles described in Section 5.1.2, this KB derives a degree of belief of .95 for "$0 \in \mathtt{running}^{*}$", which by the definition of our shorthand is equivalent to "$1 \in \mathtt{running}$". This mechanism allows the agent to make causal predictions between certainty and falsehood, a capability that standard default logic approaches do not have. The agent can use this added power to make quantitative judgments about the relative merits of different courses of action.

5.2.2 Domain objects in causal statistics

In Chapter 2, I stated that it is convenient to write causal rules that apply to an entire class of domain objects, using quantification. For example, to describe a car-starting situation that applies to all cars, a standard non-statistical approach could write:

$$\forall c[\mathtt{c\text{-}ignition}(c, m) \rightarrow \mathtt{c\text{-}running}(c, m')]$$

i.e. when *any* car's ignition is engaged it will start, as opposed to a particular car. The above rule uses car-fluents instead of regula. 'luents, due to the addition of the domain object argument c. This universal formula can be instantiated for a particular car c and a particular moment m, simply by substituting for the variables c and m.

Statistical causal rules can also quantify over classes of objects. In Section 2.4.3, I described how domain objects can be parameters to fluents using tuples of objects and moments. This notation can be applied in a manner analogous to the non-statistical causal rule above, to produce:

$$\$(\mathtt{c\text{-}running}^{*} \mid \mathtt{c\text{-}ignition}) = v$$

where v is the odds value, which can be less than certainty (unlike the non-statistical rule above). This rule also uses car-fluents, i.e. subsets of $\mathtt{cars} \times \mathtt{moments}$, and captures that the proportion of objects with their ignitions engaged (presumably just cars) that will also be running during the next moment, is v. This statistical rule is "instantiated" for particular cars and moments by constructing the degree of belief

$$\mathtt{B}(\langle \mathtt{c},\mathtt{m}\rangle \in \texttt{c-running}^* \mid \langle \mathtt{c},\mathtt{m}\rangle \in \texttt{c-ignition}) = v$$

where the "individual" is now a pair of individuals.

This tuple construction easily extends to cases where statistics are specific to subclasses of objects. For example, suppose the reasoner needed to represent a statistic that encodes the observation "Chevys[4] start only half of the time in cold weather". This is a generalization over both moments and Chevys. I represent this observation by conditioning on additional constraints, namely that the moment must be cold and the car must be a Chevy, in the following way:

$$\$(\texttt{c-running}^* \mid \texttt{c-ignition} \cap (\texttt{cars} \times \texttt{cold}) \cap (\texttt{Chevys} \times \texttt{M})) = 1 \ .$$

The first cross product generates all car-moment tuples in which the moment is cold, and the second cross product generates the car-moment tuples in which the car is a Chevy. Both constraits are intersected with the car-fluent c-ignition, to produce all those tuples where a Chevy's ignition is engaged during a cold moment. The statistical value of 1 captures that the odds are even money that such a car will be running during the next moment. Using this technique, statistics can generalize not just over moments, but any relevant domain class.

5.3 Competing Causal Statistics

This section introduces one of the main ideas of this thesis: that the reality of competing statistics is not a bug, it's a feature. In particular, competing statistical causal rules can be used to solve the representational side of the qualification problem, which proclaims the need to make predictions based on limited evidence. In statistical causality, this limited evidence is the most specific reference class implied by the current information about the objects involved in the prediction. This idea is detailed in the following sub-sections.

5.3.1 Potatoes in tailpipes

It is well known that statistically-founded beliefs such as I have described have nonmonotonic characteristics [Bacchus, 1988], i.e. the preferred statistical belief in the proposition "$\mathbf{x} \in \mathbf{a}$" can change (increase of decrease) as a result of new knowledge about $\mathbf{x}$'s inclusion in other sets. This approach handles exceptions to action success

[4] I do not intend to case aspersions on any particular manufacturer of automobiles, but if I did, it would be Chevys.

by the way that new reference classes are chosen when additional sentences are added to the KB. For example, we add the following additional sentences to the KB:

$$\texttt{\%(running}^{*} \mid \texttt{ignition} \cap \overline{\texttt{running}} \cap \texttt{potato)} = .1 \qquad (5.3)$$

$$0 \in \texttt{potato} \qquad (5.4)$$

This new KB derives the degree of belief of .1 for "1 ∈ **running**" rather than the previous degree .95.

5.3.2 Exceptions to exceptions

Ginsberg and Smith [1987], noted that a problem with McCarthy's default solution to the qualification problem is that exceptions to action success, such as potatoes in tailpipes, may themselves have exceptions, such as a special exhaust system with two tailpipes. In other words, determining the sufficient conditions for concluding that a qualification defeats an action also gives rise to the qualification problem. This is not a problem in my approach: exceptions to exceptions simply give rise to more specific reference classes. For example, suppose that we know that the car has a special dual-tailpipe exhaust system, written as the assertion "0 ∈ **dual-tailpipe**". Given the following statistic:

$$\texttt{\%(running}^{*} \mid \texttt{ignition} \cap \overline{\texttt{running}} \cap \texttt{potato} \cap \texttt{dual-tailpipe)} = .8 \qquad (5.5)$$

our degree of belief that the car runs will be based upon this new reference class. My approach simply treats exceptions (and exceptions to exceptions, *ad infinitum*) precisely in the same fashion as it treats all qualifications: as evidence for choosing the appropriate reference class.

These features capture the defeasible behavior that default logics were created to emulate, yet the underlying inferences on statistics are monotonic; the nonmonotonic component exists entirely within the assignment of degrees of belief and acceptance.

5.3.3 Causal generalizations

As I mentioned at the beginning of this chapter, a crucial feature of this use of statistics for causal reasoning is how statistics generalize over what does *not* appear in the description of the reference class. For causal reasoning, this means that fluents that do not appear in a reference class are generalized over when that reference class is used in a statistic. For example, the statistic

$$\texttt{\%(running}^{*} \mid \texttt{ignition)} = .95 \qquad (5.6)$$

embeds within it the relative impact of potatoes in tailpipes, dead batteries, and every other qualification that might defeat the car starting. For example, if it is believed that a dead battery will prevent a car from starting (all of the time). (5.6) can be used to derive the following bound:

$$\%(\overline{\texttt{charged}} \mid \texttt{ignition}) \leq .05$$

In fact, we can say something much stronger: the union of all the car-starting disabling qualifications q_i (conditioned on `ignition`) must have a statistical value less than or equal to .05. Thus causal statistics implicitly express generalizations over "what can go wrong". This solves the qualification problem.

Tenenberg and Weber [1989] present a way to formally demonstrate how statistics generalize over properties not explicitly included in the reference class, via the following theorem about conditioning on the elements of a partition:

$$\%(a \mid b) = \%(a \mid b \cap c) \cdot \%(c \mid b) + \%(a \mid b \cap \bar{c}) \cdot \%(\bar{c} \mid b)$$

which follows easily from Additivity and the Chain Rule (it can also be expressed in terms of odds, but the result is much more cumbersome). This theorem says that a statistic can be expressed as itself conditioned on some additional evidence and weighted by the statistic about the new evidence, summed with the same quantity for the negation of the evidence. This can be written in terms of potatoes and car starting in the following way (using two letter abbreviations for the properties `ignition`, `running` and `potato`):

$$\%(\texttt{ru}^* \mid \texttt{ig}) \;=\; \%(\texttt{ru}^* \mid \texttt{ig} \cap \texttt{po}) \cdot \%(\texttt{po} \mid \texttt{ig}) + \%(\texttt{ru}^* \mid \texttt{ig} \cap \overline{\texttt{po}}) \cdot \%(\overline{\texttt{po}} \mid \texttt{ig})$$

This also demonstrates how the statistical approach evaluates qualifications *in context*, e.g. the statistic about the potato is conditioned on the evidence already being considered, namely `ignition`. Recall that the lack of context sensitivity was a criticism of the default logic approach of minimizing disability (cf. Section 4.2.2).

5.4 Statistics vs. Defaults

This section compares the statistical inference and default theory representations of causal reasoning by way of an example. In case the reader is weary of cars and starting them, the scenario of this example involves factors that affect whether a certain computer is "up", i.e. executing instructions. I will describe a feature of the scenario and follow with its representation in both statistical and default causal rules, and then compare the two.

The time line will be divided roughly into minutes. The computer in question continues to run fairly reliably, going down or being taken down about once in every sixteen hours. This fact could be represented as follows, first using a statistic and then a default:

$$\$(\texttt{up}^* \mid \texttt{up}) = 1000/1 \qquad\qquad \texttt{up}(t) \leadsto \texttt{up}(t') \tag{5.7}$$

We may also know of mechanisms that may affect the state of the computer, such as the running of the shutdown program, where the computer is usually not up after shutdown has been run. This is written as:

$$\$(\mathtt{up}^* \mid \mathtt{shutdown}) = 1/100 \qquad\qquad \mathtt{shutdown}(t) \leadsto \neg\mathtt{up}(t')$$

When these rules are applied to particular moments, they lead to conflicting statistics and conflicting defaults that neither definition of specificity can resolve. However, we may know that the computer is up when the shutdown program is running, *by definition*, i.e. add the following sentences to the two approaches:

$$\mathtt{shutdown} \subset \mathtt{up} \qquad\qquad \mathtt{shutdown}(t) \rightarrow \mathtt{up}(t)$$

Therefore, in both approaches, this fact permits specificity to prefer the use of information about the shutdown program.

To see some important differences in the two approaches, recall the rules in (5.7). The default rule says that it is always reasonable to assume that the computer remains operating from one minute to the next. The statistic says the same if the acceptance mechanism declares that thousand-to-one odds are convincing enough. In many situations they will be, but suppose that the computer in question is part of an international banking system, and 1000/1 may not be sufficient to reasonably guarantee that a funds transfer has taken place. Thus unlike default logic, the use of statistics allows a context-sensitive acceptance mechanism, e.g. by incorporating aspects of decision theory to evaluate costs and therefore define "reasonable".

As I have mentioned previously, the lottery paradox highlights an important difference between the two approaches. The default that allows us to persist the operation of the computer from one minute to the next also allows us to persist the operation from hour, day or decade to the next. This is not true of the statistical approach, where the belief can "decay" over time. We can construct the frequency distribution corresponding to the number of moments that the computer stays up based on the statistic in (5.7), i.e. the function:

$$f(x) = \%(\bigcap_{i=1}^{x}\{m : m + i \in \mathtt{up}\} \mid \mathtt{up})$$

By the chain rule, we can rewrite this function definition as:

$$f(x) = \prod_{i=1}^{x} \%(\{m : m + i \in \mathtt{up}\} \mid \bigcap_{j=0}^{i-1}\{m : m + j \in \mathtt{up})$$

This can be simplified further if the fluent "up" is independent of its far past given its immediate past, i.e. for all $i > 0$:

$$\%(\mathtt{up}^* \mid \mathtt{up} \cap \{m : m - i \in \mathtt{up}\}) = \%(\mathtt{up}^* \mid \mathtt{up})$$

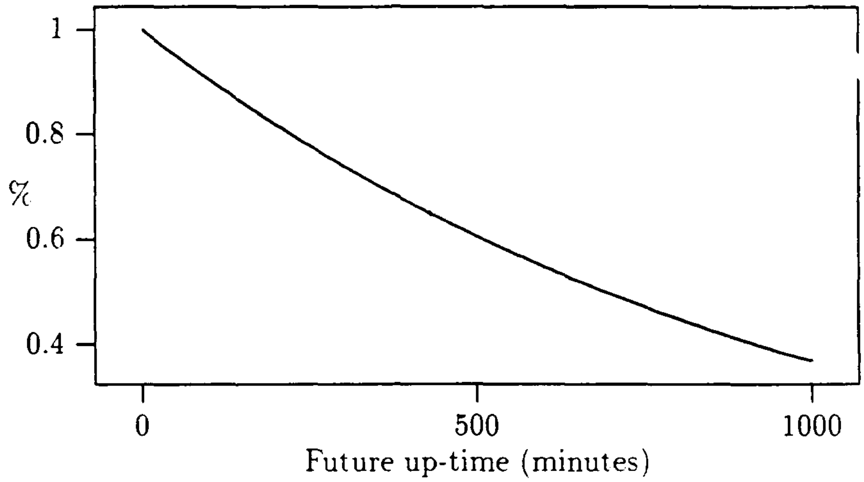


Figure 5.2: Exponential distribution of how long a hypothetical computer will remain "up", demonstrating the way that statistical persistences decay over time.

This distribution corresponds to an exponential parameterized by the frequency value corresponding to the odds given in (5.7), as shown by Figure 5.2.

Statistical rules are also more expressive even when actual numbers are not specified. For example, suppose that the computer tends to go down more often during thunderstorms. This can be represented using statistics as follows:

$$\%(\mathtt{up}^* \mid \mathtt{up} \land \mathtt{thunder}) < \%(\mathtt{up}^* \mid \mathtt{up})$$

There is no corresponding statement in default logic, since defaults (and even analytical truths) are all given the same weight. Thus default logic cannot support decisions based on the relative strength of evidence.

5.5 Survivor functions and Persistence

Dean and Kanazawa [1988] developed a statistical solution to the frame (persistence) problem by using a concept from queuing theory called a ***survivor function.*** Using our notation and terminology, these survivor functions were expressed as statistics for each fluent set p:

$$f_p(\delta) = \%(\{t : t + \delta \in p\} \mid p)$$

Intuitively, these functions derive the probability that a fluent will remain true after a given length of time δ. Dean and Kanazawa derive this function from the simpler statistic for one moment ahead, i.e. $\%(p^* \mid p) = \lambda$, as I did in the preceding section. Under certain conditions, the resulting survivor function is a geometric distribution

$f_p(\delta) = \lambda^{\delta}$ (or in the continuous case, an exponential distribution $f_p(\delta) = \epsilon^{-\lambda\delta}$). For example, suppose there is a 90% chance that your car will not be towed from an illegal parking space during a given hour. Given certain assumptions about the independence of the trials, this means that there is a 81% chance that the car will not be towed after two hours, a 73% chance after three hours, etc. Therefore, the survivor function can be used to derive a degree of belief that a fluent persists over time, solving the frame problem.

5.5.1 Survivor functions are a special case

My approach subsumes the use of survivor functions, because the statistics they embody are a special case of the statistical causal rules I have described. In fact, my approach allows for much more flexible survivor functions that are sensitive to contextual information. For example, suppose the illegally parked car above is parked next to a fire hydrant, which defines the more specific reference class in the statistic:

$$\%(\texttt{not-towed}^{*} \mid \texttt{not-towed} \cap \texttt{by-hydrant}) = .85$$

This statistic can be used to define a survivor function that incorporates this contextual information.

5.5.2 Projection rules

In addition to survivor functions, Dean and Kanazawa do allow an additional form of statistical assertion called a *projection rule*, which they use to derive a degree of belief that a fluent *changes*, e.g. in our notation:

$$\%(\texttt{running}^{*} \mid \overline{\texttt{running}} \cap \texttt{ignition} \cap \texttt{has-gas}\ldots) = x \tag{5.8}$$

In words, this statistic captures the proportion of times that the car will be running given that previously it was not running, its key was turned, it had gasoline, plus the truth of all other relevant preconditions. Projection rules are much the same as another special case of our statistical causal rules.

5.5.3 Survivor functions and competing statistics

The principal deficiency of Dean and Kanazawa's approach is that they do not handle, or even recognize, the problem of conflicting reference classes. The reference class problem appears even within their restricted statistical assertions, when representing survivor functions and projection rules for both a fluent and its negation. Information about whether a fluent will change competes with information about whether the fluent's negation will persist. For example, consider the survivor function for the negation of running:

$$\%(\overline{\texttt{running}^*} \mid \overline{\texttt{running}}) = z$$

from which it follows that:

$$\%(\texttt{running}^* \mid \overline{\texttt{running}}) = 1 - z$$

which can be applied whenever (5.8) can be applied, leading to a choice between x and $1 - z$ as the probability of a sentence such as "$1 \in$ **running**". Therefore, since they do not provide a mechanism like specificity for resolving competing reference classes, they do not solve the frame problem in a consistent way. For the same reason, they also do not address the qualification problem at all.

5.6 Chapter Summary

The mechanisms of statistical inference are similar to those of default logic, including the principle of specificity. Thus the default logic solution to the qualification problem works in statistical causal reasoning as well. However, the statistical approach to causal reasoning has the following advantages: it can avoid "lottery paradox" inconsistencies, statistics are more directly connected to empirical domain observations, and statistics are more expressive than default rules. Also, this statistical approach can represent the domain-dependent knowledge needed to infer the persistence of fluents, which is needed to solve the frame problem. A condensed version of this chapter (and the previous chapter) appears in Tenenberg and Weber [1989].

Chapter 6

Highest Impact Heuristics

The previous chapter showed how to derive a statistical probability for a prediction given knowledge about other fluent/time combinations. Taking the principle of specificity to its logical extreme. the most appropriate reference for this prediction is that which incorporates the most domain knowledge. This most specific reference class may contain many fluents which are largely or completely irrelevant to the prediction at hand. For example. by specificity, the reference class of the best statistic for car starting will include the day of the week, the color of the car, and other fluents that the agent can be said to know (or even is aware of) yet they will not significantly bear on the statistical value of the prediction.

This problem requires more than a distinction between independent and dependent preconditions; there are fluents that fall in between. Consider knowing that it is cold when starting your car. Depending, of course, on the extent of the chill and the fortitude of the car. being cold is perhaps to big a factor to be ignored, yet it is relatively unimportant compared with fluents such as the key being in the ignition, gas in the tank. etc. Thus there is an intuitive *ordering* on factors when making a prediction. e.g. `ignition` is considered before `cold` is considered before `friday`.

Considering fluents in a reasonable order can lead to valuable computational savings. In many approaches to evidence combination, the statistic for a complex reference class is computed incrementally by adding individual pieces of evidence one at a time. Although the statistical value for a reference class is the same for any order that the fluents are added, the accuracy after n fluents have been added depends on the choice of the fluents. A performance goal for statistical prediction is to achieve adequate accuracy with a relatively small n. If the most relevant knowledge is considered first, then an estimate of the accuracy may show that the computation can be curtailed.

This chapter proposes a solution to this problem for the classic Bayesian method of incrementally computing statistics. Given a set of primitive fluents, my approach incrementally builds more specific reference classes by considering these fluents in order of decreasing magnitudes of their statistical impacts. As more fluents are considered. the magnitude of the impacts are used to estimate the current error, and when this is small enough or time is critical, the computation terminates.

6.1 Bayesian Updating and Impact

A key theorem in Bayesian techniques is the following "recursive evidential updating" rule:

$$\%(h \mid r \cap f) = \%(h \mid r) \cdot \frac{\%(f \mid h \cap r)}{\%(f \mid r)}$$

which follows easily from Bayes rule. It tells us how to compute the new statistical value for a hypothesis h, obtained when a reference class r is made more specific by adding a new factor f. A version using odds is derived by dividing by the update rule for $\overline{h}$, i.e.

$$\frac{\%(h \mid r \cap f)}{\%(\overline{h} \mid r \cap f)} = \frac{\%(h \mid r)}{\%(\overline{h} \mid r)} \cdot \frac{\%(f \mid h \cap r)}{\%(f \mid \overline{h} \cap r)}$$

which by the definition of odds, gives:

$$\$(h \mid r \cap f) = \$(h \mid r) \cdot \frac{\%(f \mid h \cap r)}{\%(f \mid \overline{h} \cap r)} \tag{6.1}$$

Thus the quotient on the right-hand side provides a numerical way to update the statistical value for a reference class given a new constraint f. This quotient is very important in my approach, so it will be worthwhile to define a special term to denote it:

Definition 6.1 (Impact) *The term* $!(f,h,r)$ *is the* odds impact *(or just* impact*) of* f *on* h *given* r, *defined to be*

$$!(f,h,r) = \frac{\%(f \mid h \cap r)}{\%(f \mid \overline{h} \cap r)}$$

Therefore, (6.1) can be simplified to read:

$$\$(h \mid r \cap f) = \$(h \mid r) \cdot !(f,h,r) \tag{6.2}$$

Impact values are all non-negative numbers, as are odds values. If f is independent (cf. Section 5.1) of h given r, then $!(f,h,r) = 1$. Since the impact value is multiplied in, a value of one will not change the odds value. Impact values in the interval $[0,1)$ will decrease the odds and therefore capture a negative effect, and values from $(1,+\infty)$ increase the odds, capturing a positive effect.

6.1.1 Impact ordering

Applying (6.2) recursively derives the following theorem:

$$\$(h \mid \bigcap_{i=1}^{n} f_i) = \$(h) \cdot \prod_{i=1}^{n} !(f_i, h, \bigcap_{j=1}^{i-1} f_j)$$

In words, the statistical value for a hypothesis h conditioned on a reference class consisting of the intersection of factors f_i is equal to the product of the prior odds for h and the product of the impacts of all the factors, with each impact conditioned on the previous factors (with respect to some ordering). Since this theorem is derived from the chain rule, the resulting value is the same for *any* ordering of the factors. For example, consider deriving a statistical value for the hypothesis that the car will be running, given that the weather is very cold and the car has a block heater[1], using the following statistical impacts:

$$!(\texttt{cold},\texttt{running}^*,\texttt{M}) = .3$$

$$!(\texttt{heater},\texttt{running}^*,\texttt{M}) = 1.06$$

$$!(\texttt{cold},\texttt{running}^*,\texttt{heater}) = .85$$

$$!(\texttt{heater},\texttt{running}^*,\texttt{cold}) = 3$$

where $\texttt{M}$ is the set of all moments (an implicit condition on all causal statistics). The posterior odds for the prediction $\$(\texttt{running}^* \mid \texttt{cold} \cap \texttt{heater})$ is independent of the order that $\texttt{cold}$ and $\texttt{heater}$ are considered, as evidenced by the calculations:

$$!(\texttt{cold},\texttt{running}^*,\texttt{M}) \cdot !(\texttt{heater},\texttt{running}^*,\texttt{cold}) = .3(3) = .9$$

$$!(\texttt{heater},\texttt{running}^*,\texttt{M}) \cdot !(\texttt{cold},\texttt{running}^*,\texttt{heater}) = 1.06(.85) = .9$$

which are each multiplied by the prior odds $\$(\texttt{running}^* \mid \texttt{M})$ to derive the posterior odds for the prediction.

6.1.2 Impacts and likelihoods

In traditional Bayesian evidential updating, it is considered important that the domain is such that the new factor is independent of the current reference class given the hypothesis (or the negation of the hypothesis). When this is true, the quotient in (6.1) (i.e. the impact) is simplified in the following way:

$$\frac{\%(f \mid h \cap r)}{\%(f \mid \overline{h} \cap r)} = \frac{\%(f \mid h)}{\%(f \mid \overline{h})},$$

since by the definition of conditional independence, the reference class can be removed from the right-hand side of the conditional bar. This simplified quotient is usually called the *likelihood ratio* $\mathrm{L}(f \mid h)$ of f given h [Pearl, 1988]. Since likelihoods are not evaluated with respect to a current context (reference class) as are impacts, the

[1] A block heater is an AC device that keeps the car's engine block warm, so that the car will start in very cold weather. Contrary to what the Californians may think, Rochester rarely gets cold enough to demand such a device.

posterior odds for the hypothesis is simply the prior odds times the product of the (static) likelihoods. i.e.

$$\$(h \mid \bigcap_{i=1}^{n} f_i) = \$(h) \cdot \prod_{i=1}^{n} \texttt{L}(f_i \mid h) \ .$$

This formulation is certainly simpler, but it is not suitable for the task of causal reasoning since the conditional independence assumption between the factors precludes the interesting dynamic relationships between the impacts of fluents. In the preceding example, the example impact of `heater` is much greater when relative to the reference class for `cold`. This captures the intuition that some factors make other factors more significant, such as the way knowing it is cold increases the significance of the presence of a block heater. Similarly, notice that the impact of `cold` becomes closer to one, and therefore less significant, when made relative to the reference class `heater`. This captures the intuition that coldness matters less when starting a car if the car has a block heater. Thus dynamic interactions between fluents, such as between `cold` and `heater`, are an interesting and essential part of causal reasoning. Section 6.2.3 examines this behavior in more detail.

6.1.3 Logarithmic impact and odds

Impact and odds values are in some ways more intuitive if we use a logarithmic version of (6.1). Taking the (natural) logarithm of both sides of (6.2) and simplifying produces:

$$\begin{aligned}
\ln(\$(a \mid b \cap c)) &= \ln(\$(a \mid b) \cdot !(c, a. b)) \\
&= \ln(\$(a \mid b)) + \ln(!(c, a, b)) \\
&= \hat{\$}(a \mid b) + \hat{!}(c, a, b)
\end{aligned}$$

where the hat symbol ($\hat{}$) is used as a shorthand for logarithmic versions of both odds and impact. Because of the nature of logarithms, this new formula expresses the update of a logarithmic odds ratio as the addition of a logarithmic impact. The logarithmic version maps impacts in $[0, 1)$ to negative numbers, and impacts in $(1, +\infty)$ to positive numbers. A fluent with a negative log impact will have a negative number as an impact value. positive log impacts will have positive values, and a log impact value of zero means the fluent is independent of the prediction. Logarithmic odds also have an interesting interpretation: a log odds ratio of zero means equal weight to a fluent and its negation, a positive value means more weight to the unnegated fluent, and a negative value means more weight to the negated fluent. Also, for both log impact and odds, the positive and negative values are symmetric, i.e. $\hat{!}(c, a, b) = -\hat{!}(c, \overline{a}, b)$, and $\hat{\$}(a. b) = -\hat{\$}(\overline{a}. b)$. The relation between regular and logarithmic odds values is depicted by Figure 6.1. A belief derived from a logarithmic odds will be written in the obvious way using $\hat{\texttt{B}}(\varphi) = \ln(\texttt{B}(\phi))$.

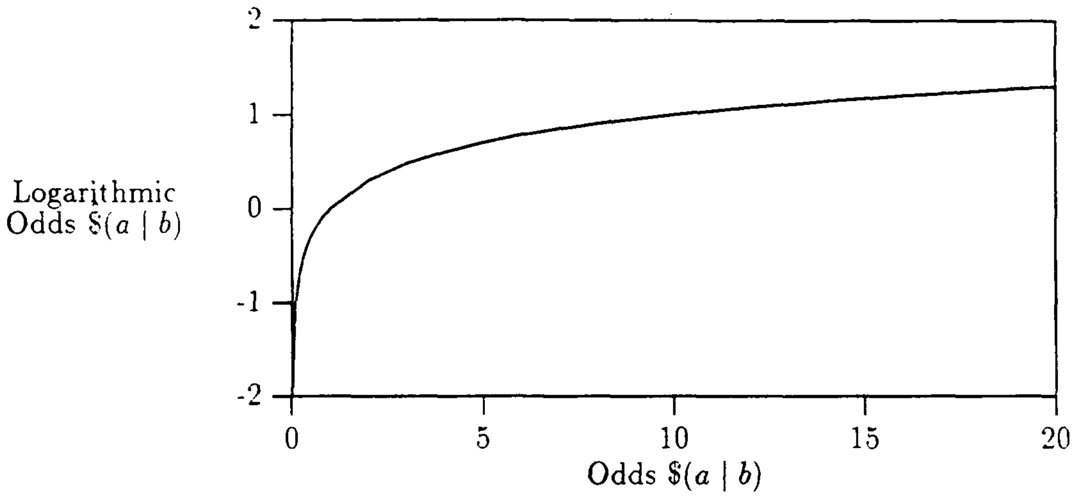


Figure 6.1: Odds values vs. logarithmic odds values. A log odds ratio of zero means equal weight to a fluent and its negation. a positive value means more weight to the unnegated fluent. and a negative value means more weight to the negated fluent.

6.2 Highest Impact First

The above discussion of impacts and their relative magnitudes suggests that fluents with high impact are more salient when deriving an appropriate reference class for a prediction; after all, fluents with zero log impact need not be considered at all, being independent. This section expresses this intuition as a heuristic ordering on the incremental inclusion of fluents in reference classes, and explores why this heuristic may promote fast convergence to a reasonable statistical value.

6.2.1 Higher impact preference

When reference classes are built incrementally, as with Bayesian updating, there will be a choice as to which order the fluents are added. This amounts choosing which statistic $\$(\mathtt{h} \mid \mathtt{r} \cap \mathtt{f}_i)$ for fluent $\mathtt{f}_i$ is best as the belief $\mathtt{B}(\mathtt{x} \in \mathtt{a})$. Specificity says little about this choice, since in general the $\mathtt{f}_i$'s will not be related by the subset relation. Impact, however, can be used to arbitrate these statistics, using the following heuristic:

Heuristic 6.1 (Higher Impact Preference) *If it is known that* $\mathtt{x} \in \mathtt{f} \cap \mathtt{g} \cap \mathtt{r}$, $\mathtt{g} \not\subset \mathtt{f}$, *and* $|\hat{\imath}(f, h, r)| > |\hat{\imath}(g, h, r)|$, *then prefer the use of* $\$(\mathtt{h} \mid \mathtt{r} \cap \mathtt{f})$ *over* $\$(\mathtt{h} \mid \mathtt{r} \cap \mathtt{g})$ *as the belief* $B(\mathtt{x} \in \mathtt{h})$.

Thus the term "highest" is used here and henceforth to mean "highest magnitude" a.k.a. "highest absolute value". The first requirement, $\mathtt{x} \in \mathtt{f} \cap \mathtt{g} \cap \mathtt{r}$, merely states that $\texttt{x}$ is known to be in all three sets. The second requirement, $\texttt{g} \not\subset \texttt{f}$, ensures that the heuristic does not conflict with a preference of the statistic using $\texttt{g}$ by specificity. Given these requirements, the fluent with the higher log impact magnitude is incorporated in the the preferred statistic.

The higher impact preference does not conflict with specificity, by definition. Instead, it complements specificity by providing a mechanism for comparing other pairs of statistics. For example, take the Nixon Diamond (cf. Section 4.5.2). Suppose that being a Republican somewhat decreases your chance of being a pacifist, as evidenced by the following statistics:

$$\$(\texttt{pacifists} \mid \texttt{people}) = 1/2$$

$$\$(\texttt{pacifists} \mid \texttt{Republicans} \cap \texttt{people}) = 2/5 \tag{6.3}$$

Where I have explicitly constrained the domain of the statistics to people and not other objects like cars and moments. Also suppose that being a Quaker greatly increases your chance of being a pacifist:

$$\$(\texttt{pacifists} \mid \texttt{Quakers} \cap \texttt{people}) = 3/1 \tag{6.4}$$

In retrospect, the impacts are as follows:

$$\begin{array}{rcccc} \hat{\imath}(\texttt{Republicans}, \texttt{pacifists}, \texttt{people}) & = & \ln(4/5) & \approx & -0.1 \\ \hat{\imath}(\texttt{Quakers}, \texttt{pacifists}, \texttt{people}) & = & \ln(6) & \approx & +0.78 \end{array}$$

The higher impact preference heuristic says that if a choice is to be made between (6.3) and (6.4) as the degree of belief $B(\texttt{Nixon} \in \texttt{pacifists})$, prefer (6.4). This is because, according to the numbers I used above, being a Quaker is more relevant to pacifism than being a Republican. Of course, if we have statistics that condition on being a Quaker *and* being a Republican, then they will be preferred (by specificity).

6.2.2 Justifying the higher impact preference

Does the higher impact preference always choose the statistic that is most indicative of the actual value of the fluent? Clearly the answer is no. Assuming that Nixon's record shows him to not be a pacifist, the negative indication of $\$(\texttt{pacifists} \mid \texttt{Republicans})$ is closer to the truth than the positive indications of $\$(\texttt{pacifists} \mid \texttt{Quakers})$, which was actually preferred.

The principle of specificity also has this problem; generalizations will be incorrect when applied to exceptions, by definition. This is why the impact and higher impact preferences are *heuristics*, which must be justified with respect to the domain. One approach would be to collect relative data from a specific domain and analyze the performance of the heuristics on the fluents of individuals. This has not been done to test specificity, and I will not do this for the higher impact preference: even if such data could be collected without bias in the sample, it would only reflect the performance in one domain.

Perhaps a psychological experiment would demonstrate that humans tend to invoke the higher impact preference. Since humans are successful reasoners in the real world, this would indicate a fact about our domain. However, the construction of such an experiment is beyond the resources and goals of this thesis.

The most compelling arguments are intuitive. If I told you that Tweety is both a bird and dead, which fact would you consider first when predicting whether she flies? Certainly the fact that she is dead bears more on the prediction. You might even ignore the fact that she is a bird if you believe that bird-ness is conditionally independent of flight given dead-ness, i.e. if dead Tweety is going to fly it's not because she has feathers. The key assumption behind the higher impact first heuristic is: it is more often true that fluents with small impacts are independent of the prediction given fluents with large impacts, than large impact fluents are independent given small impact fluents. Certainly this is true for the smallest impact of zero, since no other fluent can become conditionally independent when its evidence is added.

6.2.3 Highest impact first iteration

The higher impact preference induces an ordering on all candidate fluents that could be added to the reference class, by pair-wise ordering the resulting statistics. Thus the higher impact preference defines a *Highest Impact First* ordering for constructing incremental reference classes. For example, consider the logarithmic belief $\hat{\mathtt{B}}(0 \in \mathtt{running}^* \mid \mathrm{KB})$ where the KB contains the necessary statistical causal rules and the following knowledge:

$$0 \in (\overline{\mathtt{running}} \cap \mathtt{cold} \cap \mathtt{heater} \cap \mathtt{ignition} \cap \mathtt{damp} \cap \mathtt{friday})$$

This domain knowledge will be used incrementally, i.e. the above fluents will be considered one at a time. I will signify the fluents currently under consideration by placing assertions of set membership in place of the KB, e.g. $\hat{\mathtt{B}}(0 \in \mathtt{running}^* \mid \mathtt{running} \cap \mathtt{cold})$.

The iteration starts with the base case (using artificial values for the sake of an example):

$$\hat{\mathtt{B}}(0 \in \mathtt{running}^* \mid 0 \in \mathtt{M}) = \hat{\$}(\mathtt{running}^* \mid \mathtt{M}) = -1.06$$

using the (logarithmic) *prior odds* for the prediction, the only statistic with a vacuous reference class. The known fluents have the following logarithmic impacts:

$$\begin{array}{ll}
\hat{\mathtt{i}}(\overline{\mathtt{running}}, \mathtt{running}^*, \mathtt{M}) = -1.64 & \hat{\mathtt{i}}(\mathtt{cold}, \mathtt{running}^*, \mathtt{M}) = -0.5 \\
\hat{\mathtt{i}}(\mathtt{heater}, \mathtt{running}^*, \mathtt{M}) = +0.02 & \hat{\mathtt{i}}(\mathtt{ignition}, \mathtt{running}^*, \mathtt{M}) = +3.06 \\
\hat{\mathtt{i}}(\mathtt{damp}, \mathtt{running}^*, \mathtt{M}) = -0.7 & \hat{\mathtt{i}}(\mathtt{friday}, \mathtt{running}^*, \mathtt{M}) = +0.06
\end{array}$$

and we notice that the impact with the highest absolute value is +3.06, for the ignition being turned, `ignition`. Some of the other impacts may seem odd. In particular, the fluent for "Friday" has a positive impact. Remember, though, that so far, impacts are computed for the car running, not starting. It may actually be true that the car is driven around more on Fridays (bar-hopping, for instance). Impacts become more focused when the reference class gets larger, as we see after we add the highest impact to the prediction's prior, giving:

$$\begin{aligned} \hat{\mathtt{B}}(\in \mathtt{running}^* \mid 0 \in \mathtt{ignition}) &= \hat{\$}(\mathtt{running}^* \mid \mathtt{ignition}) \\ &= \hat{\$}(\mathtt{running}^* \mid \mathtt{M}) + \hat{!}(\mathtt{ignition}, \mathtt{running}^*, \mathtt{M}) \\ &= -1.06 + 3.06 \\ &= +2 \end{aligned}$$

and then evaluate the new impacts conditioned on the ignition being engaged:

$$\begin{aligned} &\hat{!}(\overline{\mathtt{running}}, \mathtt{running}^*, \mathtt{ignition}) = -0.05 \\ &\hat{!}(\mathtt{cold}, \mathtt{running}^*, \mathtt{ignition}) = -0.5 \\ &\hat{!}(\mathtt{heater}, \mathtt{running}^*, \mathtt{ignition}) = +0.02 \\ &\hat{!}(\mathtt{ignition}, \mathtt{running}^*, \mathtt{ignition}) = 0 \\ &\hat{!}(\mathtt{damp}, \mathtt{running}^*, \mathtt{ignition}) = -0.1 \\ &\hat{!}(\mathtt{friday}, \mathtt{running}^*, \mathtt{ignition}) = +0.01 \end{aligned}$$

Note that the impact of a fluent such as `ignition` drops to zero after it has been added. This is because it makes no difference if a fluent is added a second time; thus the Highest Impact First heuristic automatically avoids redundant consideration of fluents. Also note that the consideration of the ignition severely decreases the impact of the car not previously running. This is simply because the ignition is not often engaged when the car is already running; therefore, `ignition` covers much of the statistical impact of $\overline{\mathtt{running}}$.

The big winner of this next impact comparison is the coldness fluent, `cold`. Adding its impact produces the new belief of $2 + -0.5 = 1.5$. Incidentally, this means the believed odds of the car starting are approximately 32/1. The non-zero impacts of the remaining fluents become:

$$\begin{aligned} &\hat{!}(\overline{\mathtt{running}}, \mathtt{running}^*, \mathtt{ignition} \cap \mathtt{cold}) = -0.03 \\ &\hat{!}(\mathtt{heater}, \mathtt{running}^*, \mathtt{ignition} \cap \mathtt{cold}) = +0.4 \\ &\hat{!}(\mathtt{damp}, \mathtt{running}^*, \mathtt{ignition} \cap \mathtt{cold}) = -0.05 \\ &\hat{!}(\mathtt{friday}, \mathtt{running}^*, \mathtt{ignition} \cap \mathtt{cold}) = +0.01 \end{aligned}$$

Now that we have considered coldness, the impact of the block heater rises sharply; this is one of the most interesting features of this approach. It is due to the fact that potential components of a reference class always have their impacts evaluated with respect to the current reference class; recall that the lack of this ability was a criticism of the "minimizing disability" default logic approach (cf. Section 4.2.2). In the example above, the presence of a block heater has a large impact *only in contexts in which it is cold.* Thus fluents will often be ignored until a context (reference class) is constructed that raises their impacts. Conversely, notice how the impact of damp *diminished* when cold was considered. This is because since many cold days are also damp, the addition of cold already captures some of the statistical impact of damp; therefore, the impact of damp must scale proportionately. In these ways, the Highest Impact First heuristic leads the reasoner down paths of conditionally dependent evidence. Whereas at first it may have seemed unfortunate that impacts must be recomputed at every iteration, this fact is precisely how the reasoner can dynamically consider the relevance of fluents in context.

6.3 Highest Impact Remaining

Since these degrees of belief are computed incrementally, at each step of the iteration the reasoner has a value which can be used as the belief. The Highest Impact First heuristic will tend to make this value a better estimate (with respect to the degree suggested statistic in the KB with the most specific reference class) as more iterations are performed. In this way, the reasoner can trade speed (number of iterations) off against accuracy (specificity of the reference class). The reasoner can respond to requests for both a "quick guess" or a "slow analysis", by performing a different number of iterations.

It would be even more interesting for the reasoner to be able to decide for itself how many iterations are necessary to provide a given accuracy. That way, the reasoner can spend only as much time computing as the situation requires. This judgment is inherently contingent on the impacts of the known dependent factors, e.g. one can predict whether the sun will rise tomorrow with great accuracy in very little time, whereas it takes much longer to make a reasonable prediction about the next hockey game. Since highest impact iteration involves scrutinizing these impacts, it provides a promising framework to make the accuracy judgment.

I propose that a reasonable estimation of the error of the prediction at a given iteration is the magnitude of the highest impact over all the fluents known to contain the moment in question. If this highest impact is zero, then the current prediction is *exactly* the value of the statistic in the KB with the most specific reference class. When all impacts but the highest are zero, the absolute error is exactly that highest impact. In other cases, the highest impact is merely an estimate of the error, but it will tend to be reasonable because of the way the statistic for the current reference class generalizes over the fluents not currently included. The current statistic asserts that over all the possible states of knowledge the agent may have with respect to the fluents not yet considered, the average sum of all of the impacts will be zero. The current highest impact is an upper bound on how much the incorporation of a single fluent can defy this average.

```
function Bel(fluent h, moment m, accuracy a)
refclass = ∅
belief = prior(h)
repeat
    foreach f in known(m) compute Î(f,h,refclass)
    find g with maximum i = abs(Î(g,h,refclass))
    belief = belief + Î(g,h,refclass)
    refclass = refclass ∪ {g}
until i ≤ a
return belief
```

Figure 6.2: Algorithm for computing statistical predictions based on highest impact heuristics.

To use this heuristic, an accuracy value is attached to each request of the causal reasoner to make a prediction. The causal reasoner performs a highest impact iteration, and at the end of each iteration it examines the highest impact that was just incorporated into the prediction. If this highest impact is less than or equal to the accuracy specified, then the iteration will stop and return the current value of the posterior probability. Note that as a special case, if the accuracy is zero, then highest impact iteration will continue until all knowledge about dependent fluents is incorporated into the reference class.

Alternative related heuristics are imaginable. For example, instead of using the highest impact as an accuracy estimation, the reasoner could use the sum of *all* impacts available, or perhaps even the average impact. An important advantage of using the highest impact is that it is automatically available from the process of highest impact iteration. Thus it expends virtually no computational resources ascertaining the value of expending more computation on the prediction at hand.

6.4 Prediction Algorithm

To make the description of the computation more concrete, Figure 6.2 contains a pseudo-code version of a prediction algorithm that incorporates the Highest Impact First and Highest Impact Remaining heuristics. The algorithm is expressed as a function with three inputs: the fluent that is wished to be predicted (presumably a starred version of a primitive fluent e.g. **running***), the moment for which the prediction is to be made, and the accuracy needed for the prediction. The variable `refclass` is a set of primitive fluents whose intersection is the reference class. The variable belief will contain the floating point logarithmic odds degree of belief as it is refined during the iteration. The function call known(m) returns the set of primitive and temporally-relative fluents f such that $\mathtt{m} \in \mathtt{f}$.

When the function is invoked, refclass starts as empty (meaning the actual reference class is the set of all moments M), and the belief starts as the prior statistic %(h | M). During each iteration to follow, all impacts for the fluents known at m are computed, and the largest magnitude selected. This impact is added to the belief, and the corresponding fluent name added to refclass, thereby intersecting the fluent g stands for with the former reference class. This iteration continues until the highest impact is less than or equal to the accuracy specified.

This function is guaranteed to halt and return a value, provided that there is a finite number of fluents and the accuracy a is non-negative. This is because at worst the iteration will continue until all impacts are zero.

6.5 Fooling Highest Impact Heuristics

Again, these heuristics are not guaranteed to be optimal, and it is easy to construct such cases where the behavior is less than optimal. For instance, the Highest Impact First heuristic is fooled by impacts that cancel each other, i.e. one is the negative of the other (for logarithmic impacts). Recall the example of reasoning about starting a car while knowing that the weather is cold and damp yet the car has a block heater. At one point during the iteration, the current degree of belief was +2 and the impacts for these three fluents were as follows:

$$\begin{aligned}
\hat{\imath}(\mathtt{cold}, \mathtt{running}^*, \mathtt{ignition}) &= -0.5 \\
\hat{\imath}(\mathtt{heater}, \mathtt{running}^*, \mathtt{ignition}) &= +0.02 \\
\hat{\imath}(\mathtt{damp}, \mathtt{running}^*, \mathtt{ignition}) &= -0.1
\end{aligned}$$

leading the iteration to chose the impact of cold, changing the degree of belief to 1.5. The new impacts, relative to this new fact, became:

$$\begin{aligned}
\hat{\imath}(\mathtt{heater}, \mathtt{running}^*, \mathtt{ignition} \cap \mathtt{cold}) &= +0.4 \\
\hat{\imath}(\mathtt{damp}, \mathtt{running}^*, \mathtt{ignition} \cap \mathtt{cold}) &= -0.05
\end{aligned}$$

whereupon the impact of heater was added to derive a new belief of 1.9; assuming the impact of damp is independent of the new fact heater, it will remain at −0.05, therefore rendering a final degree of belief of 1.85. Note that this highest impact ordering of the fluents, i.e. cold, heater, then damp, is actually less optimal than the ordering damp, heater, then cold. In fact, a reasonable strategy would be to consider damp, which derives a belief of 1.9, and then ignore cold and heater altogether. This is because the impacts of cold and heater cancel each other out, being roughly the negation of one another. The Highest Impact First heuristic lead the iteration "into a dead end" by considering cold first.

This example also demonstrates how the highest impact remaining heuristic may make overly conservative estimates of the accuracy. At the beginning of the example above, the highest impact was that of `cold`, at magnitude 0.5. In fact, the current value of 2 is much closer to the final value of 1.85. A conservative accuracy is not so much a problem, since the iteration will continue and tend to derive a tighter estimate.

Of more concern are the cases where the accuracy estimate is overly optimistic, since it could cause the iteration to stop and the error will not be discovered. For example, suppose the car has four spark plugs. If one spark plug malfunctions, or perhaps even two, the car still may start most of the time. However, if *all four* plugs malfunction, then the car has very little chance of starting. This is a case where four individual fluents (corresponding to the health of each spark plug) have relatively low impacts, but the impact of the intersection of the fluents is monumental. If the highest impact remaining heuristic looks at the individual impacts, it may decide that the current prediction is accurate, when in fact, if it continued for four more iterations, the resulting belief would be drastically different.

These degenerate cases all stem from the fact that it is not always possible to assess the impact of the *intersection* of multiple fluents in terms of the impacts of the individual fluents. One solution to this problem is to add problematic intersections to the set of primitive fluents used in the iteration. For example, in addition to the primitive fluents **dead-plug1**, ..., **dead-plug4**, the reasoner can explicitly represent the fluent $\bigcap$**dead-plug**n. This latter fluent will become known for a moment m (i.e. in the KB) exactly when the four individual fluents are known during m. In such a case, it will assert itself with a large impact, and be immediately noticed by the highest impact heuristics (if it is chosen to construct the reference class, the impacts of the individual fluents will fall to zero, since they are conditionally independent of the prediction). This technique is intuitively appealing (especially to believers in holistic meaning), since there appears to be a qualitative difference between the cases where a single plug malfunctions versus when all the plugs malfunction.

The logical extreme of this technique would compute the impacts of *all* intersections of primitive fluents. This would be a ridiculous approach, since it would only take one iteration to pick the intersection corresponding to the most specific reference class, thereby precomputing the iterative computation in the impacts of the intersections. To avoid the exponential explosion of the number of impacts to be computed, intersections should be explicitly added only for problematic examples, as described above.

6.6 A Performance Solution to the Qualification Problem

Highest Impact heursitics provide a unique solution to a previously unexplored level of the qualification problem. In Chapter 4, I said that the qualification problem says that predictions must still be made despite incomplete knowledge. Later on in that chapter, I described how default logic can be used to make predictions commensurate with all the situational knowledge the agent has. In Chapter 5 I described how statistics can be used to make more robust and detailed predictions than default logic, but still with respect to *all* the situational knowledge of the agent. Highest Impact heuristics go beyond this by providing a quantitative way to ascertain what subset of the agent's situational knowledge is most *practical* to consider.

Any decision about how much reasoning is *practical* must be defined in terms of the cost of the reasoning, the quality of the result, and the demands on these values imposed by the situation. Evidential updating specifies a simple way to trade reasoning cost for accuracy: more iterations means a more specific reference class and therefore a (heuristically) better estimate. On top of this, the Highest Impact First heuristic increases the expected accuracy for a given number of iterations. And finally, the Highest Impact Remaining heuristic decreases the number of iterations performed for a given estimated accuracy.

6.7 Chapter Summary

It may not be practical for the reasoner to base a degree of belief in a prediction on *all* fluents known to contain the moment in question. The Highest Impact First heuristic defines an order of importance of these fluents, based on their relative statistical impacts on the prediction. This inspires an iterative procedure where the degree of belief is incrementally refined, allowing the reasoner to trade prediction speed for accuracy. Furthermore, the Highest Impact Remaining heuristic provides a way to estimate the current accuracy given the value of the highest impact. A condensed version of this chapter (and the next chapter) appears in Weber [1989b].

Chapter 7

Parallel Computation of Statistical Impacts

In the last chapter. I presented an algorithm that incrementally refines a degree of belief in a prediction. At each iteration. a fluent is chosen for intersection with the current reference class by examining the impacts of every fluent and picking the highest. Since these impacts are sensitive to the nature of the current reference class. they must all be recomputed at each iteration, or at least examined to see if they must be recomputed. This would seem to be a prohibitively expensive operation. It would be senseless to use an algorithm where the complexity of a single iteration rivals the entire running complexity of more "naive" algorithms.

Fortunately. this problem is solved by the power of a key insight: *the impacts for the individual fluents can be computed in parallel.* The parallel time complexity to compute the impacts is then merely the maximum time it takes to serially compute a single impact. This chapter discusses the implementation of this insight both in an abstract machine architecture, and in a program called HITEST that runs on parallel hardware.

7.1 Design of a Prediction Machine

Imagine the role of a causal prediction "server" as part of a larger reasoning system. This server accepts requests for the probabilities of fluents at particular times, e.g. "2 ∈ `running`" to within specified estimated accuracies. The probabilities returned are based on a KB of domain knowledge and statistical causal rules that are available to the server.

Now imagine that this server is not just one program, but a collection of programs communicating and running in parallel. Requests are received by a program which orchestrates the computation, and accumulates the impacts towards the incrementally refined prediction. This program communicates with a field of other programs, one for each fluent, who compute the impacts of their fluents with respect to the current reference class at each iteration. This architecture is depicted in Figure 7.1. The statistical causal rules are encoded in the fluents' prior probabilities, stored in the

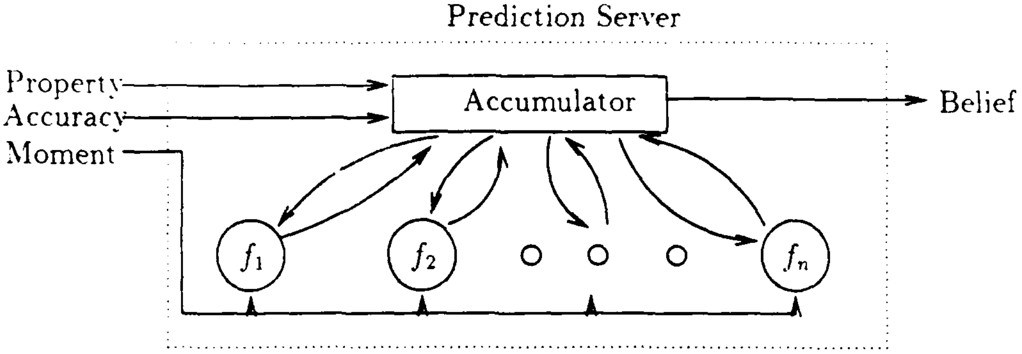


Figure 7.1: Parallel architecture within a statistical prediction server. The round nodes compute of the impact of fluent f_i on the prediction fluent with respect to the current reference class when the moment in question belongs to f_i.

central accumulator, and the reference class dependent impacts stored at the *fluent processors*. The domain knowledge is encoded by which fluent processors respond to the input moment m, i.e. exactly those processors whose corresponding fluents contain m.

7.1.1 The central accumulator

The accumulator "accumulates" the impacts into a degree of belief. The initial value of this belief is the prior probability of the input fluent h that is to be predicted, which is equal to the statistic $\%(h \mid \mathtt{M})$, where $\mathtt{M}$ is the set of all moments.

The accumulator collects incoming impacts from the fluent processors, and picks the largest impact magnitude. The winning impact is then added to the current degree of belief, implementing the Highest Impact First heuristic described in the last chapter. If the winning impact is less than or equal to the input accuracy, then the accumulator answers the original request by returning the current degree of belief, implementing the Highest Impact Remaining heuristic. Otherwise, it sends the "name" of the winning fluent processor to every fluent processor, and repeats the procedure in this paragraph. This algorithm is summarized by the pseudo-code in Figure 7.2. Note that this algorithm does not use the value of the input moment. The prior probability is the same for all input moments, since the corresponding statistic is conditioned on the entire set of moments, and the domain knowledge about the input moment is implicit in which fluent processors respond by sending impacts. In other words, the accumulator does not need to be sent the input moment since it deals with the values of statistics, which are generalizations over all moments.

```
Accumulator(fluent h, accuracy a)
belief = prior(h)
loop
    receive incoming messages <source,impact>
    find message f with largest i = abs(impact(f))
    belief = belief + impact(f)
    if i $\leq$ a then break
    broadcast source(f)
forever
return belief
```

Figure 7.2: Algorithm for the central accumulator of an abstract prediction machine.

7.1.2 The fluent processors

The fluent processors all perform the same computations, just for different fluents. They also receive the same messages from the accumulator (the last winner) in tandem. The individual impacts they produce are different, since the value $\hat{I}(f, h, r)$ can be different for each fluent f, though for the same prediction fluent h and reference class r. Thus this parallel architecture is best described as SIMD (single instruction, multiple data).

A fluent processor is enabled, i.e. allowed to send impacts to the accumulator, if the input moment belongs to the processor's fluent. The domain knowledge needed for this decision is stored at each processor. An enabled processor for fluent f starts by sending its *prior impact* $\hat{I}(f, h, \mathtt{M})$ to the accumulator. It waits until the accumulator returns the name of the winner, and then computes the new revised impact, and repeats the send-revise loop.

The impact revision process will be more efficient if it is performed incrementally. A fluent processor can be thought of as a finite state machine, whose transitions are labeled by the values of the attest highest impact winner. Each transition, or equivalently, the new state the transition points to, causes a specific impact value to be output[1]. Therefore, the state of the FSM is an encoding of the current reference class in the computation, with the start state corresponding to the vacuous reference class M. With this approach, impact revision takes only the (presumably constant) time to index to and follow a state transition. This common fluent processor algorithm is summarized in Figure 7.3. An added feature of this algorithm is that a fluent processor will update its state even if it is not enabled; this allows fluent processors

[1] Perhaps it is more appropriate to call the processors "Moore-Mealy" machines, which are FSMs with outputs added to transitions.

```
Processor(fluent h, moment m)
state = 0
loop
    if I_contain(m) then send impact(state, h)
    receive broadcast message m
    state = transition(state, h, winner(m))
forever
```

Figure 7.3: Algorithm for each fluent processor. These processors differ on the data captured by the functions I_contain. impact, and transition.

to be enabled during a computation, at which point they will immediately assert their impacts.

7.2 A Parallel Programming Environment

The University of Rochester Computer Science Department is very interested in parallel computing. Ongoing research is examining the construction of parallel programming environments and their use for specific applications. Several machines, operating systems, languages, libraries and interfaces are available to support experimentation with both simulated parallelism and actual parallelism. This section briefly describes one combination of these resources that was used to implement the algorithmic constructions of this thesis.

7.2.1 The BBN Butterfly

Bolt, Baranek and Newman designed and sells a MIMD (multiple instruction, multiple data) multiprocessor machine called the BBN Butterfly Parallel Processor. This machine consists of a small to medium number of Motorola 680x0 processors with significant amount of memory per node, communicating over an FFT-topology switch. I used a 120-node version, with one megabyte of memory per node, software floating point, and no file system (programs are loaded over the Ethernet). The primary Butterfly operating system is called Chrysalis, an object-oriented operating system with superficial kernel-call similarities to UNIX. Windowing and node-allocation software allows multi-user use of the machine with limited protection.

7.2.2 SMP

SMP is a Simple Message Passing parallel programming environment implemented under Chrysalis on the Butterfly [LeBlanc *et al.*, 1986]. This package provides C-language functions for the creation of families of heavyweight processes with fixed communication topologies. Processes communicate using explicit send and receive functions. Processes are balanced among available physical nodes, which can run several processes using context-switching.

7.3 HITEST: A Program for Statistical Causal Reasoning

HITEST (Highest Impact Techniques for Empirical STatistics) is a parallel program family that implements the Highest Impact First and Highest Impact Remaining heuristics to incrementally refine a degree of belief in a causal prediction. HITEST runs on the BBN Butterfly, creating processes that communicate using SMP library calls.

7.3.1 Command line invocation

Rather than implement a server that answers a series of requests for predictions (cf. Section 7.1), HITEST creates a family of processes to answer a single request. HITEST is invoked from the Butterfly shell using a command line of the form:

```
hitest target accuracy [fluents ...]
```

where *target* is the fluent that is to be predicted (actually the prediction is for *target**, i.e. that fluent over the next moment), *accuracy* is an upper-bound on the highest impact that may be ignored, and *fluents* is a list of fluents known over the current moment, each prefixed by a "+" or "-", depending on whether the moment is known to belong to the fluent or its negation, respectively. For example, the command line

```
hitest running 0 -running +ignition
```

would start a computation of the statistic

$$\hat{\%}(\texttt{running}^* \mid \overline{\texttt{running}} \cap \texttt{ignition})$$

which is equal to the posterior probability

$$\hat{\texttt{B}}(\texttt{m} \in \texttt{running}^* \mid \texttt{m} \in \overline{\texttt{running}} \cap \texttt{ignition})$$

for the current moment m. Accuracy values of greater than zero cause HITEST to compute a heuristic estimate of these values.

7.3.2 Accumulator code

The HITEST program does three things: parses the command line arguments, spawns the processes corresponding to the known fluents, and accumulates the sum of the highest impacts until the desired accuracy is reached. The accumulator code is very similar to Figure 7.2: messages are received from the fluent processors and serially examined as they arrive to determine the winner. The winning impact is added to the current belief, and the name of the winner is sent to all fluent processors.

The accumulator waits only a limited amount of time for impact messages. After the allotted time, the accumulator chooses the highest impact thus far and continues. This "timeout" feature has two advantages: the accumulator need not know how many fluents are connected to it, and individual fluent processors may die or take too long yet the computation will still progress at proper speed. Both advantages would be important in a large implementation of HITEST, with many information sources. This strategy, however, requires a more complicated message protocol, in order to avoid late messages arriving at the accumulator and reporting obsolete impacts. This is implemented by adding a field to messages that relates the iteration number to which the message pertains. The accumulator ignores impact messages for old iterations, and the fluent programs ignore winner messages for old iterations.

7.3.3 Fluent FSMs and canonical reference classes

The fluent programs invoked by HITEST are implementations of the finite-state machines discussed earlier in this chapter. Each program consists of a table of impact values and a small driver routine (the same driver is linked into each fluent program). Each program contains the impact values for both a fluent and its negation: the program will never have to respond for both fluents since the current moment cannot belong to both. When the fluent program is invoked, it is sent whether it should respond as the fluent or its negation, as well as the name of the fluent to be predicted.

The fluent program starts in state zero (the vacuous reference class), sends the impact of this state to the accumulator, and then waits for a response from the accumulator as to the winner. The transition to a new state is computed from the current state and the name of the winner, and the process continues.

One approach to the implementation of the transitions is to construct a table of new states with three dimensions corresponding to the current state, the latest winner, and the fluent being predicted. This way, the transition function from the algorithm in Figure 7.3 would be simply a table lookup. This is not in fact how the transition function is implemented in HITEST fluent programs. Instead, the transition function is a closed form expression using the values for the three table dimensions. It is simple enough to assign integers to some arbitrary ordering of fluents, but this approach also requires a canonical ordering of the states corresponding to reference classes.

Since each fluent can be known to contain. known not to contain. or not known to contain or not contain a given moment. a set of n primitive fluents will generate 3^n different descriptions of intersections corresponding to reference classes. Given the fluents $f_1, \ldots, f_n$. HITEST fluent programs use the following canonical ordering:

$$\emptyset, \overline{f_1}, \overline{f_1} \cap \overline{f_2}, \ldots, f_1, f_1 \cap \overline{f_2}, \ldots, \bigcap_{i=1}^{n} f_i$$

The ordinal number of each reference class corresponds to a state number. Therefore. given s. the number of the current state, w the number of the last winning primitive fluent. and v. whether the current moment belongs to w or its negation (1 or 0. respectively). the next state t is computed by:

$$t = s + 3^w \cdot (1 + v)$$

This expression obviates the need for an explicit transition table. Thus the fluent programs require only a table of impacts for each combination of a state. the fluent being predicted. and whether the program's fluent or its negation is known.

Although the above approach using a canonical ordering of reference classes is simple and does not require a transition table, there is a disadvantage that will make it impractical for large numbers of fluents. This disadvantage is that it cannot exploit the savings in the number of states that stems from conditional independence information. For example, if fluent f is conditionally independent of g given h and r. then the two impacts $\hat{I}(f, h, r)$ and $\hat{I}(f, h, r \cap g)$ will be identical. This means that with respect to fluent f's impact on the prediction h, the two reference classes r and $r \cap g$ are identical. and do not require separate states. An explicit transition table can simply wire the transition on winner g while in state r back to state r: a large number of conditional independence assumptions would soften the exponential blowup of reference classes. It is not possible to eliminate states when using a canonical ordering of reference classes. However, since the representation and exploitation of conditional independence information is beyond the scope of this thesis (but see Pearl [1988]). HITEST uses the canonical ordering approach in the interests of simplicity and worst-case space complexity.

7.3.4 HITEST examples

This section contains three example runs of the HITEST program. They are, of course, based on the scenario of starting ones car when faced by factors such as cold, damp. weather and the presence of a block heater (recall that a block heater helps keep the engine warm when the car is not running).

Figure 7.4 shows the results from a request to know the degree of belief that the car will be running to an accuracy of 0. given that the car is currently not running. the

```
[2] hitest running 0 -running +ignition +cold +damp +heater
Prediction is now -1.935520
                +cold        -.139392
                +ignition    4.733500
                +damp        -.037231
                +heater      .001621
                -running     -1.590289
        +ignition has highest impact 4.733500
Prediction is now 2.797979
                +cold        -.765457
                +damp        -.314173
                +ignition    -.000000
                +heater      .037387
                -running     -.465519
        +cold has highest impact -.765457
Prediction is now 2.032521
                +ignition    .000000
                +damp        -.254750
                +heater      .331753
                +cold        .000000
                -running     -.779762
        -running has highest impact -.779762
Prediction is now 1.252759
                +cold        .000000
                +damp        -.196710
                +ignition    -.000000
                -running     -.000000
                +heater      .514899
        +heater has highest impact .514899
Prediction is now 1.767658
                +cold        .000000
                +damp        -.301324
                +ignition    -.000000
                +heater      -.000000
                -running     -.000000
        +damp has highest impact -.301324
Prediction is now 1.466333
                +ignition    .000000
                +heater      .000000
                +damp        .000000
                +cold        .000000
                -running     .000000
Iteration complete with final prediction of 1.466333
Elapsed time: 1.525624 seconds
```

Figure 7.4: Example run of HITEST program. The fluents are considered one at a time in an order determined by the dynamic values of the fluents' impacts.

ignition is engaged, the weather is cold and damp, and a block heater is present. The accumulator spawns the processes corresponding to these fluents, and then starts the iterative refinement of the degree of belief. At the start of each iteration, the accumulator prints out the current belief, which starts out as the prior probability of `running`. This value, like all other values manipulated by HITEST, is a (natural) logarithm. Thus the (non-logarithmic) prior odds starts as approximately $\epsilon^{-2} \approx 1/7$, i.e. the car is running one-seventh of the moments. The data used to derive the priors and impact values is artificial; see Section 7.3.6 for more information on the derivation of these values.

The doubly-indented lines following the "`Prediction`..." line contain the impacts for the individual fluent programs. Notice that `ignition` jumps out as the winner, with `running` as a distant second. After `ignition` is chosen to be added, the belief jumps from (non-logarithmic) 1/7 to 16/1. This belief reflects that the car starts fairly reliably, and also the fact that the car will also generally remain running when the ignition is engaged. In the next iteration, notice that the fact that the car isn't currently running is a negative influence on the car subsequently running, although not as much as it was before `ignition` was considered. The fluent `cold`, however, is the winner in this iteration, since considering `ignition` amplified the impact of `cold`. This reflects the fact that `cold` interferes with the car's ignition more than it interferes with the continued running of the car.

In the third iteration, the impact of `heater` has risen by an order of magnitude, since the presence of a block heater only becomes relevant when the weather is known to be cold. The impact of `damp` has become less negative after the consideration of `cold`, since many cold days are also damp days. Both impacts, however, are dominated by the impact of the fact that the car is not running. The impact of `heater` is incorporated into the belief in the fourth iteration, and the impact of `damp` in the fifth. When all fluents have been considered, and therefore all impacts have fallen to zero, the desired accuracy of zero is reached and the iteration stops. The final prediction of 1.466 says that the odds are approximately 13/3 in favor of the car starting in this scenario.

Figure 7.5 contains another example HITEST run. This time, the weather is known to *not* be cold. The interesting feature of this example is how the impact of `heater` is small but positive during the first three iterations, because it *might* be cold outside, but when the fact that it is not cold is explicitly considered, the block heater has no impact on the prediction that the car will be running.

Figure 7.6 illustra es how HITEST applies the Highest Impact Remaining heuristic by interrupting the iteration when the highest impact falls below .3. This point is reached in the third iteration, when the highest impact is found to be only .224 for `damp`. The accumulator stops the execution of the fluent programs, and returns a value of 2.56. The actual error, as evidenced by the impact that `heater` would have had on the next iteration, is less than .1, well within the accuracy desired.

```
[2] hitest running 0 -running +ignition -cold +heater
Prediction is now -1.935520
                -cold        .009906
                -running     -1.590289
                +ignition    4.733500
                +heater      .001621
        +ignition has highest impact 4.733500
Prediction is now 2.797979
                +ignition    .000000
                +heater      .037387
                -running     -.465519
                -cold        .078690
        -running has highest impact -.465519
Prediction is now 2.332459
                +ignition    .000000
                +heater      .071792
                -cold        .135637
                -running     .000000
        -cold has highest impact .135637
Prediction is now 2.468096
                +ignition    .000000
                -running     .000000
                -cold        .000000
                +heater      .000000
Iteration complete with final prediction of 2.468096
Elapsed time: .843312 seconds
```

Figure 7.5: Another example run of HITEST program. Note how the fluent **+heater** (the car has a block heater) has little impact in the beginning, and when **-cold** (the weather is not cold) is considered the impact of **+heater** falls to zero, since it is only relevant when the weather is cold. Thus **+heater** is never considered.

```
[2] hitest running .3 -running +ignition -damp +heater
Prediction is now -1.935520
                -running     -1.590289
                -damp        .012499
                +ignition    4.733500
                +heater      .001621
        +ignition has highest impact 4.733500
Prediction is now 2.797979
                +heater      .037387
                +ignition    .000000
                -running     -.465519
                -damp        .128041
        -running has highest impact -.465519
Prediction is now 2.332459
                +ignition    .000000
                -running     .000000
                -damp        .224135
                +heater      .071792
        -damp has highest impact .224135
Iteration complete with final prediction of 2.556595
Elapsed time: .693687 seconds
```

Figure 7.6: One more example run of HITEST program. This time the fluent `+heater` is not considered because it always remains below the accuracy threshold.

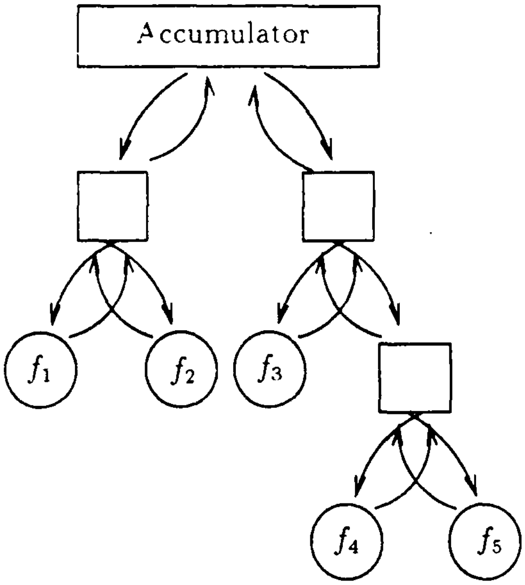


Figure 7.7: A tree topology for HITEST, where the accumulator is at the root and fluent programs at the leaves. This allows for impact comparisons to be computed in parallel.

7.3.5 Parallel comparison of impacts

The "star" communication topology of Figure 7.1 is actually a special case of the topology used in HITEST. HITEST builds a tree of communicating processes, where the accumulator is at the root and the fluent programs are at the leaves, as in Figure 7.7. The maximum number of children for any node in this tree is called the *fanout*. A star topology is a special case when the fanout is greater than or equal to the number of primitive fluents.

The use of a multi-level tree allows the search for the maximum impact to be conducted in parallel. Recall that in the previous discussion, the accumulator processes the incoming impact messages serially, spending time proportional to the number of fluents in finding the highest impact. In the tree topology, the accumulator need only spend time proportional to the fanout, which can be a small constant. This is because the interior nodes of the tree are also receiving the impact messages from their children and passing on the highest. Thus the highest impact is decided in parallel, for greater computational savings.

A price for parallel computation of the highest impact is an increase in the time it takes for messages to travel from the fluent programs to the accumulator. This difference is only a factor of $\log_f n$, where f is the fanout and n is the number of fluents. A more significant price may be that the entire family sends an exponential number of messages rather than a linear number. Depending on the actual communications hardware used for the message passing, this could generate a substantial amount of message collision. I did not find this to be a problem in my experiments.

7.3.6 Building statistics from data

A fluent program's impact table is what distinguishes it from other fluent programs. This table contains the impact of the program's fluent on every "future" fluent, with respect to every possible reference class. For n fluents the dimensions of this table are n by 3^n by 2. The last factor of two comes from the fact that a fluent program knows the impacts of both the fluent and its negation (one is not derivable from the other, although they must satisfy a consistency constraint).

Even ignoring the tedium of constructing these tables by hand[2], subtle errors in the table values can make the impact assignments *inconsistent*. The constraint between the impact of a fluent i_1 and the impact of its negation i_2 is as follows: there must exist two real numbers x and y such that $i_1 = x/y$ and $i_2 = (1-x)/(1-y)$. In addition, there is a constraint on impacts of different fluents that stems from the fact that the posterior probability is the same for any order of consideration of the fluents, i.e.

$$\$(h \mid r \cap f \cap g) = \$(h \mid r \cap f) \cdot !(g, h, r \cap f) = \$(h \mid r \cap g) \cdot !(f, h, r \cap g)$$

or expressed purely in terms of impacts:

$$!(f, h, r) \cdot !(g, h, r \cap f) = !(g, h, r) \cdot !(f, h, r \cap g)$$

Thus the impacts contain some redundancy, both within a single fluent program and between fluent programs. This makes it very difficult to ensure consistency in hand-coded tables.

The program FILLERUP (Fabricated Impacts Locally ... forget it) program performs the off-line compilation of these impact tables. The input to FILLERUP is an array of *weighted outcomes* each of which contains a *before part*, an *after part*, and a *weight*. The before and after parts are each boolean assignments to the set of primitive fluents, i.e. an encoding of what was true during a moment m and its successor m'. The weight expresses how many times that particular "confluence of fluents" was observed[3]. For example, Figure 7.8 shows a portion of the outcome data used in the examples from the preceding section. Most of these outcomes deal with situations where the car starts when the ignition is engaged. The second through fourth outcomes are weighted much more than the last four outcomes to reflect the relative infrequency of (significantly) cold weather.

[2] Believe me, it is tedious. I nearly went out of my mind hand-coding an example with only *two* fluents!

[3] Thanks to my officemate Jack Veenstra for suggesting the use of weights.

```
/*   r i c d h    r i c d h */
  { {0,0,1,1,1}, {1,0,1,1,1},   0},
  { {0,1,0,0,0}, {1,0,0,0,0}, 450},
  { {0,1,0,0,1}, {1,0,0,0,1}, 450},
  { {0,1,0,1,0}, {1,0,0,1,0}, 140},
  { {0,1,0,1,1}, {1,0,0,1,1}, 140},
  { {0,1,1,0,0}, {1,0,1,0,0},  26},
  { {0,1,1,0,1}, {1,0,1,0,1},  28},
  { {0,1,1,1,0}, {1,0,1,1,0},  10},
  { {0,1,1,1,1}, {1,0,1,1,1},  13},
```

Figure 7.8: A portion of the outcome data used by the FILLERUP program to build impact tables. Each line contains the before and after values of the fluents `running`. `ignition`. `cold`. `damp`. and `heater`. plus the number of moments this outcome was observed.

FILLERUP scans each outcome. enumerating through the 2^n "before" reference classes represented by each outcome (all subsets of the given fluent values). For each fluent that is true in the "before" and "after" data, FILLERUP adds the weight to the cardinality of the set which is the intersection of the fluent with the reference class. This is also done for the sets that are intersections of the reference class and pairs of true fluents. one "before" and one "after". The resulting cardinalities. by the cardinal definition of relative frequency (cf. Section 5.1), are sufficient to compute all combinations of impacts. In addition. the priors can be computed by simply totaling the weights of outcomes in which the fluent is true "after". and dividing by the total of all weights.

In a real sense. the FILLERUP program derives statistical causal rules from domain observations (simulated observations, in my examples). This removes the burden of specifying informative and consistent causal rules from the knowledge engineer. Though FILLERUP is not a very complex program, this is quite an innovative approach to the origin of causal rules. This feature further demonstrates the power of a statistical approach to causal reasoning. The derivation of causal rules is rarely even discussed in non-statistical approaches to causal reasoning, largely due to the disparity between empirical observations and constructions like default rules.

The cardinality values can be computed by scanning the outcome data in a single pass. This means that the cardinalities, and therefore the impacts they characterize, can be incrementally computed as new outcome data arrives. Though HITEST and FILLERUP do not attempt to do this, it is straightforward to construct a system that "trains" on domain observations, even when intermixed with requests for predictions. This approach is reminiscent of the work on the encoding of relative frequencies for evidential reasoning about classes in connectionist networks [Shastri, 1985].

Ideally, the input to the FILLERUP program would be actual data from some interesting domain. Unfortunately, I am still looking for a source for data that would support interesting experiments. Some ideas I had were: the stock market (or economic indicators), success or failure of purchasing soda from a machine, and sports statistics [Loui, 1987b]. None of these ideas ended up being practical. I hope that when the ideas of this thesis are embedded in some larger project, such data will be accessible.

7.4 Chapter Summary

Highest Impact Iteration becomes a practical approach when the statistical impacts can be computed in parallel. This suggests a parallel architecture where individual asynchronous units compute relevant impacts and send these values to a central node that accumulates the highest impacts for each iteration. HITEST is a working program on the BBN Butterfly that performs this incremental refinement of statistical beliefs from impacts computed in parallel. The priors and impacts for this computation are compiled from real or simulated empirical observations. A condensed version of this chapter (and the previous chapter) appears in Weber [1989b].

Chapter 8

Propagating Causal Uncertainty

An intuitive objection to the use of conditional statistics for causal reasoning is that when deriving a degree of belief in a prediction, the evidence or KB must consist entirely of practical certainties. For example, when predicting "$1 \in \texttt{running}$", in order to use the statistic

$$\%(\texttt{running}^* \mid \texttt{ignition} \cap \texttt{cold}) = \texttt{p} \tag{8.1}$$

the reasoner must have "$0 \in \texttt{ignition}$" and "$0 \in \texttt{cold}$" as practical certainties. How can the reasoner derive a degree of belief in "$1 \in \texttt{running}$" if the sentence "$0 \in \texttt{cold}$" is not a practical certainty, but instead believed with some degree q? This question is addressed in the following sections.

8.1 Incorporating Indirect Evidence

A quick answer is that (8.1) is the wrong statistic; the correct reference class incorporates the reference class that was used to derive the degree of belief in "$0 \in \texttt{cold}$", instead of $\texttt{cold}$ itself. For example, suppose that the certainty "$-1 \in \texttt{snowing}$" was used to derive the degree of belief for "$0 \in \texttt{cold}$". This knowledge can be used directly in deriving a belief for "$1 \in \texttt{running}$", using the value of the statistic:

$$\%(\texttt{running}^* \mid \texttt{ignition} \cap \{m : m - 1 \in \texttt{snowing}\}) = \texttt{q}$$

i.e. here I have incorporated the temporally-relative fluent (cf. Section 2.4.2) that is the set of moments such that it was snowing during the previous moment. Incorporating all the evidence used in deriving the degree of belief for "$0 \in \texttt{cold}$" covers the effect of that value on the degree of belief in "$1 \in \texttt{running}$".

This approach generalizes recursively to handle the case where "$-1 \in \texttt{snowing}$" is believed to a non-certain degree: simply use the reference class that incorporates the knowledge used in deriving this latter degree of belief. Degrees of belief must ultimately ground out in certainties, or as Pearl [1988] puts it, the posterior probability of a random variable depends only on its prior probability and the values of *instantiated* variables.

Although this solution is representationally sufficient, it is not likely to be practical in automated reasoning. The statistics required must include reference classes over combinations of all temporally-relative fluents ("snowing two moments ago", "snowing three moments ago", etc.), which is much greater than the presumably already large set of primitive fluents. It would be ridiculous to require an explicit statistic that conditions car starting on the words in the weather report from the previous day. This grossly violates the practical stipulation that causal rules should only involve *direct* influences.

8.2 Virtual Evidence and Reasoning by Cases

Suppose we are simply given a degree of belief for "0 ∈ `cold`", without access to the particular evidence used in deriving it. This could be for practical reasons, as described above, or the evidence may have even been forgotten. In such cases, the evidence for the degree of belief in a direct influence is said to be *virtual*. In general, virtual evidence is not suitable for deriving a degree of belief in some other fact. For example, suppose I tell you that I have evidence that leads me to a 5/1 odds that Tweety is a bird: this would seem to be enough to derive a high degree of belief that Tweety flies. However, the evidence I actually used could involve that Tweety lives in Antarctica (lots of penguins live down there). I also could have used evidence including that Tweety lives in Australia (lots of parakeets live down there). These two alternatives for evidence imply very different probabilities that Tweety flies.

Certain conditional independence assumptions are necessary to utilize virtual evidence [Pearl, 1988]. Let h be the fluent being predicted, let f be the fluent considered to be a "direct" influence, and let ϵ be the intersection of the temporally-relative fluents that comprise the virtual evidence that was used to derive the degree of belief in f. We would like to derive the statistical value for $\%(h \mid \epsilon)$ in terms of more accessible statistics concerning the statistical relationships between h and f, and between f and ϵ. To do so involves the following theorem for *case-based* reasoning:

$$\%(h \mid \epsilon) = \%(h \mid \epsilon \cap f) \cdot \%(f \mid \epsilon) + \%(h \mid \epsilon \cap \overline{f}) \cdot \%(\overline{f} \mid \epsilon) \tag{8.2}$$

This formula expresses conditional statistics in terms of the disjoint and exhaustive cases for another fluent and its complement. The statistical value for $\%(f \mid \epsilon)$ is equal to a degree of belief we are given for f, which is also equal to $1 - \%(\overline{f} \mid \epsilon)$. If we assume that the the evidence ϵ influences h ***only through*** f, i.e. ϵ is conditionally independent of h given f, (8.2) simplifies to:

$$\%(h \mid \epsilon) = \%(h \mid f) \cdot \%(f \mid \epsilon) + \%(h \mid \overline{f}) \cdot (1 - \%(f \mid \epsilon)) \tag{8.3}$$

This is the relationship we have been looking for: the statistic for the prediction conditioned on virtual evidence can be derived from the statistic for the direct influences conditioned on this evidence. and the statistic for the prediction conditioned on the direct influences. For the cold car-starting example above. we would write:

$$\begin{aligned}
&\%(\texttt{running}^{*} \mid \texttt{ignition} \cap \{m : m-1 \in \textbf{snowing}\}) \\
&\quad = \%(\texttt{running}^{*} \mid \texttt{cold}) \cdot \%(\texttt{cold} \mid \{m : m-1 \in \textbf{snowing}\}) + \\
&\qquad \%(\texttt{running}^{*} \mid \overline{\texttt{cold}}) \cdot (1 - \%(\texttt{cold} \mid \{m : m-1 \in \textbf{snowing}\})) \\
&\quad = \%(\texttt{running}^{*} \mid \texttt{cold}) \cdot \%(\texttt{cold}^{*} \mid \textbf{snowing}) + \\
&\qquad \%(\texttt{running}^{*} \mid \overline{\texttt{cold}}) \cdot (1 - \%(\texttt{cold}^{*} \mid \textbf{snowing}))
\end{aligned}$$

The statistic on the left-hand side is needed to assign a degree of belief to the sentence “$1 \in \texttt{running}$”. This formula shows how to derive that statistical value in two steps: first to derive the degree of belief for “$0 \in \texttt{cold}$” conditioned on the knowledge “$-1 \in \textbf{snowing}$”. and then apply it to “$1 \in \texttt{running}$”.

8.3 Bayesian Networks

Case-based reasoning about direct influences is the basis of *Bayesian networks* [Pearl. 1988. Pearl. 1985]. Each node in these networks represents a random variable. and also a computational unit in charge of computing the posterior probability for its random variable given the values of other nodes in the network. These distributed computations are performed using local updates. where each node only has access to information from its immediate neighbors, corresponding to the direct influences described above. The update rule is based on the theorem in (8.3), but more complex because of the interactions *between* multiple direct influences. Instead of merely two cases for a single influence in (8.3) for f and $\overline{f}$, multiple influences require an exponential number of disjoint and exhaustive cases for each combination of factors or their negations, i.e. for n direct influences:

$$\%(h \mid e) = \sum_{u_i \in \{\overline{f}_i, f_i\}} \underbrace{\%(h \mid \bigcap_{j=0}^{n} u_j)}_{a} \cdot \underbrace{\%(\bigcap_{j=0}^{n} u_j \mid e)}_{b}$$

This is problematic in three ways:

1. the node must compute the joint distribution for each combination of values for its direct influences given the virtual evidence (b above),

2. the node must know the exponential (in n) number of statistics for itself conditioned on the possible values of the direct influences (a above), and

3. the node must sum an exponential number of terms to compute the result.

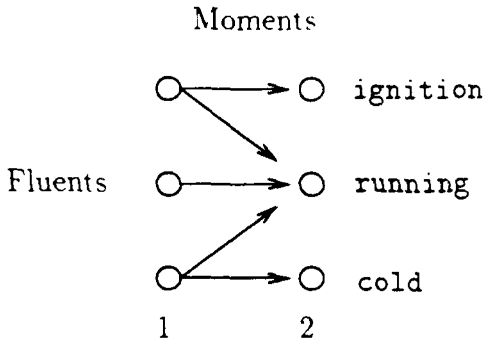


Figure 8.1: A piece of a causal reasoning network. Fluents at moment 1 are direct influences on themselves at moment 2. plus `ignition` and `cold` at moment 1 are direct influences on `running` at moment 2.

Again. conditional independence assumptions can help. If the direct influences are independent of each other (except through h). then the joint distribution of direct influences (problem 1) is simple to compute locally as the product of the individual conditional statistics. In terms of networks (graphs), this assumption amounts to requiring that the nodes are *singly connected*, i.e. there is at most one path for evidence to flow between any two nodes.

Pearl [1988] also showed how a model of canonical interactions between the influences and the hypothesis. based on independence assumptions about exceptions. can be used to reduce the number of statistics needed (problem 2) and also the number of terms in the summation (problem 3). Thus statistical reasoning can be very efficient in domains that support the independence assumptions needed.

8.3.1 Temporally-Staged Networks

Unfortunately, it appears that causal reasoning[1] is not well-suited to these independence assumptions. Dean and Kanazawa [1989] have pursued the application of Bayesian networks to causal reasoning, and have subsequently rejected the assumption that nodes are singly connected.

To see why, consider the fragment of a causal reasoning network in Figure 8.1. The nodes correspond to the random variables "1 $\in$ `ignition`", "2 $\in$ `ignition`", ..., "2 $\in$ `cold`". All three fluents at moment 1 are direct influences on themselves at moment 2. i.e. whether the car is currently running is a direct influence on whether the car will be running subsequently. In addition, the nodes for `ignition` and `cold` at moment 1 are direct influences on `running` at moment 2. When these causal connections are duplicated in order to span several moments (done by Dean and

[1] I.e. explicit reasoning about how fluents change over time, as opposed to the more general way that Pearl uses the term "causal" to mean influences between random variables.

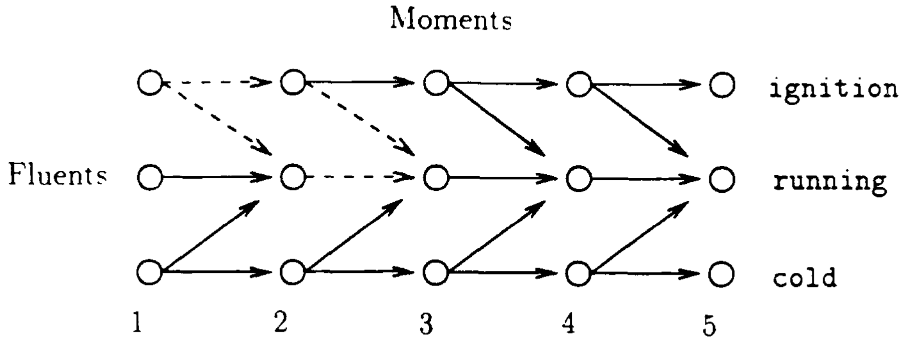


Figure 8.2: Several stages of a temporally-staged network. Note that a loop has formed (dashed lines), rendering the network multiply-connected.

Kanazawa [1989]) as in Figure 8.2, the network is definitely not singly-connected. One such loop is shown by the dashed lines. These loops occur whenever fluents at one moment are allowed to directly *or* indire·tly influence themselves at the next moment. Given that this appears to be a fundamental feature of causal reasoning, it is still an open problem to provide a scheme for efficient propagation of uncertainty on temporally-staged networks. Solving this problem is beyond the scope and goals of this thesis: however, the following section presents an interesting approach that may provide a solution.

8.4 Clustered Causal Reasoning Networks

Pearl [1988] describes a technique for eliminating loops in causal networks called *clustering*. This technique combines selected nodes in a network, producing a new network which is singly-connected. For example, to eliminate the dashed parallelogram from the network in Figure 8.2, we could combine the nodes for "1 $\in$ **ignition**" and "2 $\in$ **ignition**", and also combine the nodes for "2 $\in$ **running**" and "3 $\in$ **running**". This would produce a pair of nodes with a single connection between them. These new nodes can then pass information and compute posterior probabilities as described before, except that the random variables they represent have *four* values corresponding to the values of the combinations of the two possible values for the two nodes from which they were created. In general, a clustered node will have n^m possible values if it was created by combining m other nodes each with n possible values. The update process must compute the posterior probabilities for each of these n^m values, and pass these values on to its neighbors.

To eliminate all loops due to the influence of a fluent at one time on itself in the future, we can cluster all nodes for each fluent over time This produces a network with one node for each fluent f which incorporates the nodes in the original network representing the random variables "$m \in f$" for each moment m. The clustered version

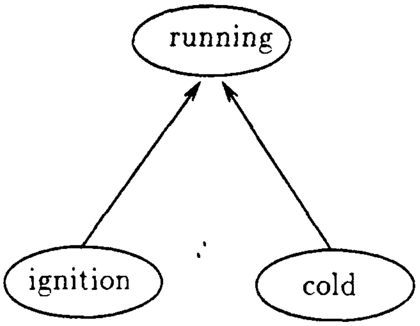


Figure 8.3: A version of Figure 8.2 where all nodes over time for each fluent are clustered into single nodes. This operation eliminates loops caused by fluents influencing themselves in the future.

of Figure 8.2 is depicted in Figure 8.3. In general, the result is not guaranteed to be singly-connected[2], since this depends on the particular causal connections between fluents. However, now that the loops between the same fluent have been eliminated, it is more feasible to find domains for which the clustered network is singly-connected.

The clustered network is also more realistic than the temporally-staged network as a topology for distributed causal computations. In the staged version, the same causal connections are duplicated between all columns of nodes for each moment. This is not only an inefficient use of connections, but it also is not suitable for a model that learns or corrects causal connections, since these changes would have to be repeated for each moment. In addition, all fluents are explicitly represented at all moments, and the number of computational units bounds the number of moments that can be reasoned about. In the clustered version, there is only one network connection for a given causal connection, making it more realistic for the dynamic nature of causal theories. Furthermore, the computational unit corresponding to a fluent can sparsely represent the values of that fluent over time, depending on which fluent/moment pairs are considered relevant. This can also be extended to allow the network to reason about an arbitrary number of moments.

Unfortunately, the combinatorial difficulties of updating such large-scale clustering are apparent. Even if we limit the network to reasoning about a finite number of moments, say 100, then the nodes will computing and communicating distributions of 2^{100} values. I believe, however, that the situation is not so bad. In causal reasoning,

[2] It is interesting to note that clustering can actually *create* loops. Suppose that in the original network with nodes A through E, A is connected to B which is connected to C, and A is also connected to D which is connected to E. There are no loops in this network. However, if we cluster nodes C and E into a new node CE, there are two paths from A to CE, one through B and the other through D. For causal reasoning, a loop in the clustered network corresponds to a loop in the staged network only iff the clustered loop contains the same number of forward going and backward going connections. This footnote appears only for those readers really interested in this sort of thing.

evidence about one fluent at a particular moment will tend to affect other fluents only for a limited number of other moments. Information that it is raining today will only affect my belief in fluents for today, tomorrow, and perhaps yesterday. That is, updates due to new information will be largely temporally local. When a fluent node makes such local changes, it need only send the *changes* to its neighbors, instead of the entire distribution over all moments. The receiving node may then apply these local changes to produce local changes to the knowledge about its own fluent, and pass those changes on.

8.5 Chapter Summary

Causal statistics can also be used to derive degrees of belief from other degrees of belief instead of certainties. This derivation is performed by reasoning about the conditional statistics given all possible values of the variables involved. In general, this computation is combinatorially explosive, but the presence of particular conditional independencies allows for efficient distributed computation. It is an open problem whether this is possible for network-based causal reasoning, though a promising approach applies clustering techniques to make efficient computation more feasible.

Chapter 9

Conclusions

The traditional approach to the frame problem has been to look for some domain-independent constraints on causal reasoning situations that infer the persistence of fluents by default. This approach is fraught with philosophical and technical problems that question its fundamental soundness. Instead, the reasoner can succinctly axiomatize the domain-dependent information needed to infer that a fluent persists, and reason about persistences in the same manner as changes.

A causal reasoner will not have complete knowledge about a realistic domain. Solving the qualification problem involves making causal predictions in the face of this ignorance. Default rules provide a mechanism for making reasonable predictions from limited information, thereby addressing the qualification problem. However, since default rules are forced to round "usually" to "always" valuable information is lost and inconsistencies may develop in the resulting theory.

The mechanisms of statistical inference are similar to those of default logic, including the principle of specificity. Thus the default logic solution to the qualification problem works in statistical causal reasoning as well. However, the statistical approach to causal reasoning has the following advantages: it can avoid "lottery paradox" inconsistencies, statistics are more directly connected to empirical domain observations, and statistics are more expressive than default rules. Also, the statistical approach can represent the domain-dependent knowledge needed to infer the persistence of fluents, which is needed to solve the frame problem.

It may not be practical for the reasoner to base a degree of belief in a prediction on *all* fluents known to contain the moment in question. The Highest Impact First heuristic defines an order of importance of these fluents, based on their relative statistical impacts on the prediction. This inspires an iterative procedure where the degree of belief is incrementally refined, allowing the reasoner to trade prediction speed for accuracy. Furthermore, the Highest Impact Remaining heuristic provides a way to estimate the current accuracy given the value of the highest impact.

Highest Impact Iteration becomes a practical approach when the statistical impacts can be computed in parallel. This suggests a parallel architecture where individual asynchronous units compute relevant impacts and send these values to a central node that accumulates the highest impacts for each iteration. HITEST is a working program on the BBN Butterfly that performs this incremental refinement of statistical beliefs from impacts computed in parallel. The priors and impacts for this computation are compiled from real or simulated empirical observations.

Causal statistics can also be used to derive degrees of belief from other degrees of belief instead of certainties. This derivation is performed by reasoning about the conditional statistics given all possible values of the variables involved. In general, this computation is combinatorially explosive, but the presence of particular conditional independencies allows for efficient distributed computation. It is an open problem whether this is possible for network-based causal reasoning, though a promising approach applies clustering techniques to make efficient computation more feasible.

Chapter 10

Future Work

There are several interesting directions to go from this thesis. This chapter outlines a few of the more obvious ones.

10.1 Improvements

This first category of future work is improvements that can be made on the solutions this thesis has proposed. All of these improvements were already mentioned in the thesis as they became relevant, but it is worth reviewing them again.

10.1.1 Confidence intervals

Usually, an agent must derive its statistics about a class from knowledge about only a sub-class. For example, my statistic that captures "most birds fly" is based on my limited exposure to birds. I have certainly not been exposed to all birds, most birds, or even most species of birds. Therefore, my statistic about the proportion of birds that fly may actually be *way* off. How accurate my statistic is depends on the size of my *sample* and the way that those samples were selected.

If the sample size and bias (e.g. the samples are random elements of the population) are known, then *confidence intervals* can be constructed that approximate the proportion of the entire population. These confindence intervals are probabilities that the actual population proportion falls within a certain interval[1]. For example, based on my experiences with birds, I may be 95% sure that the actual proportion of birds that fly falls within the interval [.85, .95]. An interval can be constructed for any dersired probability; the greater the probability, the wider the resulting interval (e.g. 99% sure of [.83, .97]).

If the reasoner has a uniform level of acceptance, say that sentences with probability $\geq .95$ are practical certainties, then sentences such as the following

$$\%(\texttt{fliers} \mid \texttt{birds}) \in [.85, .95]$$

[1] These probabilities follow from the knowledge about the sample size and bias plus statistics about *sampling*, in the same way that other probabilities follow from knowledge about the objects and statistics about object classes.

will become practical certainties. These can be used to derive interval probabilities that particular birds (such as Tweety) are in the set of fliers. using the techniques of Kyburg [1987] and Loui [1987b].

As I mentioned in Section 5.1.3. the interval approach is a proper generalization of the point-valued statistics and probabilities used in this thesis. In other words. this thesis does not exploit any information the agent may have about the sample size and bias. It is clear that doing so is not just a matter of performing impact calculations on both the upper and lower probability bounds. and treating the results as bounds on the impact – the bounds quickly diverge to provide uninformative intervals. There is more information in the *distribution* of confidence intervals for different probabilities, which can be used to construct tighter bounds. The particulars of the use of interval probabilities in computing (and ordering!) impacts is a topic for future work.

10.1.2 Virtual evidence

The subject of Chapter 8 was the derivation of beliefs (statistically-founded probabilities) from *other beliefs*. rather than certain knowledge. This is motivated by a desire to support local computations and statistics. i.e. to explicitly represent only "direct" statistical influences between fluents. It is an open problem whether the efficient singly-connected Bayesian networks of Pearl [1988] can be used for causal reasoning [Dean and Kanazawa. 1989]. However, Section 8.4 discussed a promising approach that clusters the random variables for a single fluent over time. which may allow for efficient propagation of belief. The investigation of this approach is a topic for future work.

There is an interesting related pursuit. In singly-connected Bayesian networks. the general update rule for a given node (random variable) has a complexity exponential in n, the number of direct causal influences. Even for modest values of n, the update computation may become prohibitively expensive. Pearl [1988] describes an approach to this problem that involves additional conditional independence assumptions about the causes. A different approach is to compute an approximation of the actual update using a small subset of the direct influences.

Highest Impact Iteration seems ideally suited for this latter approach. Each direct influence computes its impact on the given node, and the highest is used as the single direct influence for the purpose of updating the node's posterior probability. This first approximation can be computed and propagated to other nodes quickly. Then the impacts are recomputed given the chosen influence, and an additional influence is chosen. and so on. The Highest Impact Remaining heuristic can also be employed to decide that the node need not consider additional influences, allowing the node to stabilize on the current posterior probability (unless, of course, the probabilities of its direct influences change). This same process is performed on all nodes, producing a network that quickly arrives at a first approximation, then gradually refines its values. The details of this idea is a topic for future work.

10.1.3 Learning and using conditional independence

The statistical causal reasoners this thesis describes require a great deal of statistical knowledge. The reliability and consistency of this knowledge will be greater if the statistics are learned from actual domain observations, rather than compiled by knowledge engineers, not to mention the cost of such a hand-compilation. In Section 7.3.6, I described a technique and a program that precompiles the impacts needed for each fluent, given data about the outcomes of trials. It even appears straightforward to update the impacts incrementally as new data becomes available. In this way, the reasoner can *learn* the impacts, and therefore the statistical causal rules, directly from interactions with the domain.

However, explicitly learning and storing all the $n^2 3^n$ impacts (for n fluents) will not be feasible for even a modestly large n. To deal with large numbers of fluents, the reasoner must discover and use the conditional independence relations that are (hopefully) abundant in the domain (cf. Section 7.3.6). By the definitions of conditional independence (Definition 5.1) and impact (Definition 6.1), we find:

$$\mathrm{I}(h, f, r) \equiv (\,!(h, f, r) = 1) \equiv (\,!(f, h, r) = 1)$$

That is, two sets h and f are conditionally independent given a third set r if and only if their mutual impacts given r are equal to one (their log impacts are zero). Thus the more conditional independence relationships the reasoner knows, the fewer impacts it needs to represent. Broad classes of independence relations can be efficiently represented using graphs, e.g. Markov random fields [Kanal, 1981] and Bayesian networks [Pearl, 1988].

These conditional independence relationships must themselves be learned. There must be more to the method than having the reasoner notice when impacts are trivial; this would require that statistics be gathered and stored for a while before that decision could be made, plus there must be some way to retract an independence assumption. Perhaps independence assumptions can be communicated effectively between agents, or perhaps they can be ported between domains through some analogy mechanism. The pursuit of these mechanisms is a topic for future work.

10.2 Planning

This thesis has discussed principles and algorithms for causal reasoning, being primarily concerned with how to predict the future. The obvious next step is to figure

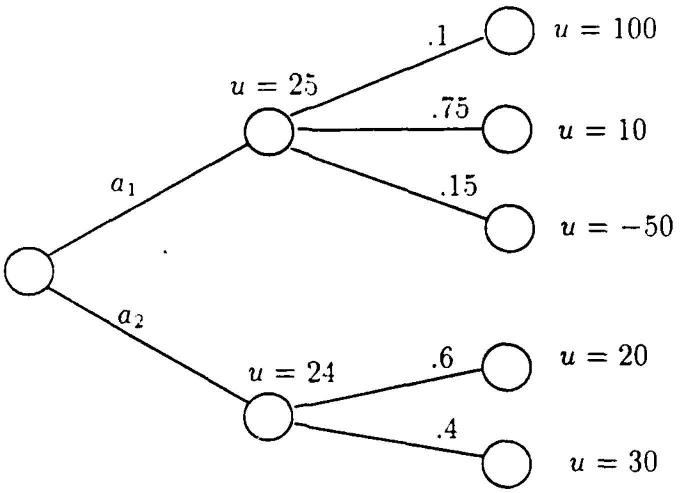


Figure 10.1: An example decision tree. The first level correponds to the two choices of action, a_1 or a_2; the second level shows the different states the world may be in after an action is performed, with probabilities for each outcome. These states have associated utilities which can be "backed up" to calculate an expected utility for each action.

out *what to do about it.* The joint process of predicting and acting is usually called *planning.*

Decision Theory is the study of techniques for making optimal action choices [Chernoff and Moses, 1959]. The standard approach requires a quantitative measure of *utility* over the different states of the domain. This utility function depends on the character, strengths, and interactions of the goals of the reasoning agent. Given the ability to predict the resulting state of the domain for different action choices, the agent can identify the optimal choice by examining the *expected* utilities of the resulting states, and picking the largest.

For example, see the *decision tree* in Figure 10.1. The agent has two action choices, and as far as it knows the first action can lead to three different states, and the second action two. It can calculate the expected utility of performing each action by the sum of the possible utilities, weighted by its probabilistic belief in each state outcome. In this case, the expected utility of action a_1 (25) is higher than that of action a_2 (24), making a_1 a better choice. This tree can be extended recursively by adding additional levels corresponding to additional action choices, backing the expected utility through all levels, thereby defining an optimal *path* through the space of decisions.

Traditional planning can be viewed as a special case of decision theory, where there is but a single outcome for each action with a weight of one. The planner chooses the action(s) that best satisfy its goals, as reflected by the utility of the outcome. Multiple weighted outcomes offer many advantages over the traditional approach, as discussed in the scant but important work on the importance of decision theory in AI [Feldman and Sproull, 1977, Loui, 1987b, Hartman, 1989].

Thus decision analysis involves two main components: the assignment of utilities to domain states, and the assignments of weights to an action's possible outcome states. This thesis has shed some light on the latter, for applications of decision theory to causal reasoning. There is, however, an important difference between the assignment of probabilities to states, for decision analysis, and the assignment of probabilities to set-membership sentences, as I have described in this thesis. Certainly an exhaustive assignment of probabilities to states generates an exhaustive assignment of probabilities to sentences, and vice versa; it is more difficult to apply partial information about sentence probabilities to evaluating the expected utility of actions. One approach is to define action outcomes to be classes of states, with utilities corresponding to the mean utility over the class: the probabilities of sentences would be used to generate these classes.

In summary, by proposing techniques for statistical causal reasoning, this thesis has also made progress on statistical planning. Much work remains to be done, primarily concerning the application of decision theoretic techniques that evaluate the utility of actions using probabilistic predictions. The incorporation of these techniques, the investigation of effective search strategies for the resulting decision trees, and the relationship of utilities to goals, are all topics for future work.

References

[Allen. 1984] James F. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23(2):123–154, July 1984.

[Bacchus *et al.*, 1989] Fahiem Bacchus. Josh Tenenberg. and Hans Koomen. A non-reified temporal logic. In *Proceedings of the First International Conference on the Principles of Knowledge Representation and Reasoning*. Morgan Kaufman, May 1989.

[Bacchus, 1988] Fahiem Bacchus. *Representing and Reasoning with Probabilistic Knowledge*. PhD thesis. University of Alberta, Fall 1988.

[Cheesman, 1988] Peter Cheesman. An inquiry into computer understanding. *Computational Intelligence*. 4:58–66, 1988.

[Chernoff and Moses, 1959] H. Chernoff and L. Moses. *Elementary Decision Theory*. New York: Wiley, 1959.

[Dean and Kanazawa, 1988] Thomas Dean and Keiji Kanazawa. Probabilistic temporal reasoning. In *Proceedings of AAAI-88*, pages 524–528, August 1988.

[Dean and Kanazawa, 1989] Thomas Dean and Keiji Kanazawa. A model for reasoning about persistence and causation. working paper, January 1989.

[Elgot-Drapkin *et al.*, 1987] Jennifer Elgot-Drapkin, Michael Miller, and Donald Perlis. The two frame problems. In Frank M. Brown, editor, *The Frame Problem in Artificial Intelligence: Proceedings of the 1987 Workshop*. Morgan Kaufman, April 1987.

[Etherington, 1987] David W. Etherington. More on inheritance hierarchies with exceptions: Default theories and inferential distance. In *Proceedings of AAAI-87*, pages 352–357, 1987.

[Feldman and Sproull, 1977] Jerome A. Feldman and Robert F. Sproull. Decision theory and artificial intelligence II: The hungry monkey. *Cognitive Science*, 1:158–192, 1977.

[Fikes and Nilsson, 1971] Richard E. Fikes and Nils J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:198–208, 1971.

[Georgeff, 1987] Michael P. Georgeff. Actions, processes, and causality. In *Proceedings of the 1986 Workshop on Reasoning about Actions and Plans*, 1987.

[Ginsberg and Smith, 1987] Matthew L. Ginsberg and David E. Smith. Reasoning about action II: The qualification problem. Technical report, Knowleq Systems Laboratory, Stanford University, May 1987.

[Ginsberg, 1986] Matthew Ginsberg. Counterfactuals. *Artificial Intelligence*, 30:35-80, 1986.

[Ginsberg, 1987] Matthew Ginsberg. Possible worlds planning. In *Proceedings of the 1986 Workshop on Reasoning about Actions and Plans*, 1987.

[Goldman, 1970] Alvin Goldman. *A Theory of Human Action*. Princeton University Press, Princeton, NJ, 1970.

[Goodwin, 1988] Scott D. Goodwin. Reasoning in temporal domains: Dealing with independence and unexpected results. In *Proceedings of CS/CSI*, pages 46-52, June 1988.

[Hanks and McDermott, 1986] Steve Hanks and Drew McDermott. Default reasoning, nonmonotonic logics, and the frame problem. In *Proceedings of AAAI-86*, August 1986.

[Hartman, 1989] Leo Hartman. *Decision Theory and the Cost of Planning*. PhD thesis, Computer Science Department, University of Rochester, 1989.

[Haugh, 1987] Brian Haugh. Simple causal minimizations for temporal persistence and projection. In *Proceedings of IJCAI-87*, pages 218-223, June 1987.

[Hayes, 1971] Patrick Hayes. A logic of actions. In *Principles for Designing Intelligent Robots*, pages 495-519. Metamathematics Unit, University of Edinburgh, 1971.

[Kanal, 1981] L. N. Kanal. Markov mesh models. In Azriel Rosenfeld, editor, *Image Modelling*, pages 239-243. New York: Academic Press, 1981.

[Kautz and Allen, 1986] Henry A. Kautz and James F. Allen. Generalized plan recognition. In *Proceedings of AAAI-86*, August 1986.

[Kautz, 1986] Henry A. Kautz. The logic of persistence. In *Proceedings of AAAI-86*, August 1986.

[Kyburg, 1970] Henry E. Kyburg, Jr. Conjunctivitis. In M. Swain, editor, *Induction, Acceptance and Rational Belief*. Reidel, 1970.

[Kyburg, 1974] Henry E. Kyburg, Jr. *The Logical Foundations of Statistical Inference*. Reidel, 1974.

[Kyburg, 1983] Henry E. Kyburg, Jr. The reference class. *Philosophy of Science*, 50:374–397, 1983.

[Kyburg, 1987] Henry E. Kyburg, Jr. Full beliefs. *Theory and Decision*, 1987.

[Kyburg, 1989] Henry E. Kyburg, Jr. Beyond specificity. Proceedings of IJCAI-89, 1989.

[LeBlanc *et al.*, 1986] Thomas J. LeBlanc, Neal M. Gafter, and Takahide Ohkami. SMP: A message-based programming environment for the BBN butterfly. Butterfly Project Report 8, University of Rochester Computer Science Department, July 1986.

[Lifschitz, 1986] Vladimir Lifschitz. Pointwise circumscription: Preliminary report. In *Proceedings of AAAI-86*, August 1986.

[Lifschitz, 1987] Vladimir Lifschitz. Formal theories of action: Preliminary report. In Frank M. Brown, editor, *The Frame Problem in Artificial Intelligence: Proceedings of the 1987 Workshop*. Morgan Kaufman, April 1987.

[Loui, 1987a] Ronald P. Loui. Response to Hanks and McDermott: Temporal evolution of beliefs and beliefs about temporal evolution. *Cognitive Science*, 11, 1987.

[Loui, 1987b] Ronald P. Loui. *Theory and Computation of Uncertain Inference and Decision*. PhD thesis, University of Rochester Computer Science Department, September 1987.

[McCarthy and Hayes, 1981] John McCarthy and Patrick J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In Bonnie Lynn Webber and Nils J. Nilsson, editors, *Readings in Artificial Intelligence*, pages 431–450. Tioga, 1981.

[McCarthy, 1977] John McCarthy. Epistemological problems of artificial intelligence. In *Proceedings of IJCAI-77*, pages 1034–1044, Cambridge, MA, 1977.

[McCarthy, 1980] John McCarthy. Circumscription–a form of non-monotonic reasoning. *Artificial Intelligence*, 13(1,2):27–39, April 1980.

[McCarthy, 1984] John McCarthy. Applications of circumscription to formalizing common sense knowledge. In *Proceedings of the AAAI Non-Monotonic Workshop*, pages 295–324, October 1984.

[McDermott, 1982] Drew McDermott. A temporal logic for reasoning about processes and plans. *Cognitive Science*, 6:101–155, 1982.

[Morgenstern and Stein, 1988] Leora Morgenstern and Lynn Andrea Stein. Why things go wrong: A formal theory of causal reasoning. In *Proceedings of AAAI-88*. pages 518–523. August 1988.

[Nute, 1986] Donald Nute. LDR: A logic for defeasible reasoning. Technical Report ACMC Research Report 01–0013, Advanced Computational Methods Center. University of Georgia, Athens, Georgia,,1986.

[Pearl, 1985] Judea Pearl. Fusion, propagation and structuring in Bayesian networks. In *Presented at the Symposium on the Complexity of Approximately Solved Problems*, April 1985.

[Pearl, 1988] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufman. 1988.

[Pelavin and Allen, 1986] Richard Pelavin and James F. Allen. A logic for planning in temporally rich domains. *IEEE transactions*, August 1986.

[Pelavin, 1988] Richard Pelavin. *A Logic for Planning in Temporally Rich Domains*. PhD thesis. Computer Science Department, University of Rochester, 1988.

[Reiter, 1980] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13(1,2):81–132, April 1980.

[Sandewall, 1988] Erik Sandewall. Non-monotonic entailment for reasoning about time and action - part I: Sequential actions. working paper, August 1988.

[Schubert, 1989] Lenhart Schubert. Solving the original frame problem without frame axioms or nonmonotonicity. In Henry Kyburg and Ron Loui, editors, *Selected papers from the 1988 Society for Exact Philosophy Conference*, 1989.

[Shastri, 1985] Lokendra Shastri. *Evidential Reasoning in Semantic Networks: A Formal Theory and its Parallel Implementation.* PhD thesis, Computer Science Deparment. University of Rochester, September 1985.

[Shoham, 1986] Yoav Shoham. Chronological ignorance. In *Proceedings of AAAI-86*, August 1986.

[Shoham, 1987] Yoav Shoham. Temporal logics in AI: Semantical and ontological considerations. *Artificial Intelligence*, 33(1):89–104, 1987.

[Shoham, 1988] Yoav Shoham. *Reasoning about Change: Time and Causation from the Standpoint of Artificial Intelligence.* The MIT Press, Cambridge, MA, 1988.

[Tenenberg and Weber, 1989] Jósh D. Tenenberg and Jay C. Weber. A statistical solution to the qualification problem (and how it also solves the frame problem). Technical report, University of Rochester Computer Science Department, May 1989.

[Tenenberg, 1988] Josh Tenenberg. *Abstraction in planning.* PhD thesis, Computer Science Department, University of Rochester, 1988.

[Touretzky, 1984] David S. Touretzky. Implicit ordering of defaults in inheritance systems. In *Proceedings of AAAI-84*, pages 322–325, 1984.

[Weber, 1988] Jay C. Weber. A versatile approach to action reasoning. Technical Report 237, Computer Science Department, University of Rochester, March 1988.

[Weber, 1989a] Jay C. Weber. The myth of domain-independent persistence. presented at the First International Workshop on Human and Machine Cognition, Topic: The Frame Problem, May 1989.

[Weber, 1989b] Jay C. Weber. A parallel algorithm for statistical belief refinement and its use in causal reasoning. Proceedings of the 1989 International Joint Conference on Artificial Intelligence, August 1989.

[Williams, 1986] Brian C. Williams. Doing time: Putting qualitative reasoning on firmer ground. In *Proceedings of AAAI-86*, pages 105–112, 1986.

[Winslett, 1988] Marianne Winslett. Reasoning about action using a possible models approach. In *Proceedings of AAAI-88*, pages 89–93, 1988.